Escuela Técnica Superior de Ingeniería
Grado en Ingeniería Electromecánica

Trabajo Fin de Grado

Desarrollo e implementación de una red de sensores para monitorización basada en ZigBee

Juan Manuel Márquez Flores

Tutor: Luis Orihuela Espina

Sevilla, julio de 2020

A Claudia, por su apoyo constante.

A Julio, por ayudarme en todo momento.

A Inma, por ser mi voz.

A Luis, por haber sido un buen tutor y por todos sus consejos.

A mis abuelos, por ayudarme económicamente y por inculcarme la disciplina.

A mis padres, por enseñarme a:

"Buscarme la vida"

Resumen

A lo largo de este documento se describe una red de sensores basada en Raspberry Pi 3 donde el procesamiento y subida de datos a la nube para su monitorización se consigue gracias a la comunicación de módulos de radiofrecuencia Xbee que mediante el protocolo inalámbrico ZigBee transmite datos, leídos por sensores, de una placa a otra. Por lo que, se plantea un prototipo para futuras aplicaciones IoT, tanto en docencia como en investigación.

La gran capacidad de computación de la Raspberry Pi permite: analizar, modificar, modelar y procesar un volumen de datos superior a otros ordenadores de placa única o microcontroladores. Además, el uso de Xbee mejora las capacidades inalámbricas que trae de base la propia placa.

Este proyecto muestra de qué manera se realiza las conexiones de los sensores y los módulos Xbee a las Raspberry Pi, la configuración de cada uno de los componentes, y la subida de datos a Thingspeak (un servidor online que permite la subida, visualización y tratamiento de los datos).

Por último, incluye un apartado donde se muestran posibles aplicaciones y líneas de investigación a seguir en un futuro para mejorar el sistema inicial.

Palabras clave: Sensores, Raspberry Pi 3, Xbee, ZigBee, Internet de las Cosas, Thingspeak

Abstract

Throughout this document describes a sensor network based on Raspberry Pi 3 where the processing and uploading of data to the cloud for monitoring is achieved through the communication of Xbee radio frequency modules that through the wireless ZigBee protocol transmits data, read by sensors, from one board to another. Therefore, a prototype is proposed for future IoT applications, both in teaching and research.

The great computing capacity of the Raspberry Pi allows: to analyze, modify, model and process a higher volume of data than other single board computers or microcontrollers. In addition, the use of Xbee improves the wireless capabilities of the board itself.

This project shows how the connections of the sensors and the XBee modules to the Raspberry Pi have been achieved, the setting of each of the components, and the uploading of data to ThingSpeak, an online server that allows data to be uploaded, visualized and treated.

Finally, a section has been included where possible applications and lines of research to be followed in the future to enhance the initial system are presented.

Keywords: Sensors, Raspberry Pi 3 Model B, XBee, ZigBee, The Internet of Things, ThingSpeak

Índice

Índice de figuras

Índice de tablas

1. Introducción

En un mundo cada vez más interconectado y en donde el incremento de nuevas tecnologías de comunicación permite la transmisión de datos prácticamente al instante, cobra más sentido aún el concepto de “Internet de las cosas” o IoT.

Conforme a Tschofenig, Arkko, Thaler, y McPherson (2015), el término "Internet de las cosas" denota una tendencia en la que un gran número de dispositivos integrados emplean los servicios de comunicación proporcionados por los protocolos de Internet. Muchos de estos dispositivos, a menudo llamados "objetos inteligentes", no son operados directamente por los seres humanos, sino que existen como componentes en edificios o vehículos, o están dispersos en el entorno.

En los últimos años, se ha convertido en una de las tecnologías más importantes del siglo XXI, permitiendo interaccionar con objetos de uso diario tales como electrodomésticos, automóviles, termostatos, sistemas de vigilancia, siendo posible la comunicación fluida entre cosas, personas y procesos. De esta forma, los objetos físicos pueden enviar y recopilar datos con apenas intervención humana mediante informática de bajo coste. (Oracle, 2020)

Según la visión de Microsoft (Microsoft 2020), el uso de esta tecnología permite a las empresas realizar mantenimiento predictivo, un seguimiento más fidedigno de la eficiencia de sus cadenas de producción, supervisión de los equipos de detección y solucionar problemas de forma remota; siendo todas ellas áreas con gran repercusión en los ingresos.

Por consiguiente, incluir un sistema IoT en una organización permite mejorar los tiempos de respuesta y gestionar de una forma ligera y eficaz los procesos de conservación de productos. Por citar un caso, se podría regular de manera automática la humedad y temperatura de una estancia en donde se preservan

productos, con la finalidad de facilitar su conservación. (Mendieta, Herrera, & Jimenez Peña, 2019)

1.1. Motivación

La primera idea del proyecto consistía en montar una red de sensores para monitorizar diversas características en los laboratorios de ingeniería de la escuela donde se usarían varias Raspberry Pi desplegadas en los diferentes espacios, a cada una de las placas se conectarían un grupo de sensores cuya finalidad sería medir ciertas variables, para a su vez enviar toda la información a una única Raspberry Pi utilizando el protocolo ZigBee, y a través de un monitor conectando en esta última, mostrar unas gráficas de los resultados.

Sin embargo, debido al cese de actividades y al confinamiento provocado por el estado de alarma establecido para combatir la pandemia del coronavirus. Se tuvo que adaptar el espacio, convirtiendo la residencia habitual del estudiante en el laboratorio a ser monitorizado.

Así pues, el proyecto tiene como objetivo el desarrollo e implementación de una red de sensores mediante el protocolo ZigBee, y su volcado de datos a un servidor web donde, de una forma elegante, se muestran los datos para su visionado y consulta, sin necesidad de estar físicamente en el lugar, lo que permite disponer de un sistema replicable y escalable para futuros proyectos en el ámbito de la ingeniería y afines.

Para el alumno, realizar un Trabajo de Fin de Grado de esta índole supuso ser gratificante a la vez que atractivo. El hecho de enfrentarse a un desafío de verdad y no a una simulación, aprender y aplicar conocimientos de electrónica, computadores, redes y programación. Y, sobre todo, la sensación satisfactoria de terminar el Grado con algo que montó el mismo y es funcional.

1.2. Objetivos

El objetivo principal del proyecto es el montaje de un sistema IoT, en donde una Raspberry Pi actuará como coordinador central y la otra como nodo. Aprovechando la gran capacidad de computación de las Raspberry Pi que permiten tanto la lectura de diversos sensores en cada una de ellas, como la transmisión por radiofrecuencia de los datos recogidos para ser monitorizados, evitando así el uso de otros sistemas de envío de datos como el 4G, que precisa de un coste adicional a través de una compañía de telecomunicaciones. O, permitiendo un alcance mayor que el permitido por Bluetooth y WiFi.

Este objetivo principal se puede dividir en algunos objetivos específicos:

- Utilizar sensores analógicos, ya que de base la Raspberry Pi tan solo permite digitales.
- Profundizar en los tipos de conexiones y protocolos que permite el uso de Xbee, y elegir el que mejor se adapte al proyecto.
- Buscar una forma atractiva de visualizar los datos que no dependa de un sistema físico, como permite Thingspeak.
- Desarrollar una memoria accesible a cualquier compañero para poder usarla como base o ayuda para otros proyectos de investigación.

1.3. Estructura

En este subcapítulo se introduce un breve resumen de cada uno de los capítulos que componen el documento, con el objetivo de ayudar en la lectura y comprensión.

- Capítulo 2: Estado del arte.

  En este apartado se realiza un análisis sobre el contexto actual de cada una de las tecnologías que intervienen en el presente documento, y aquellas afines de uso en el presente.

- Capítulo 3: Tecnologías empleadas.

En este capítulo se explora en profundidad cada una de las herramientas utilizadas en el proyecto, con idea de resolver cualquier duda que pudiera surgir más adenlante en la lectura.

- Capítulo 4: Desarrollo del proyecto.

  Sección en donde se expone cada uno de los pasos que se han realizado durante la ejecución del trabajo. A fin de simplificar su exposición, los pasos siguen una distribución que permite replicar el propio proyecto a la par que se realiza la lectura.

- Capítulo 5: Conclusiones.

  En el último capítulo se detalla un resumen de los conocimientos que el alumno ha adquirido y usado durante la realización del proyecto, así como aquellos desafíos que se encontrado y cómo se han solventado, además de las mejoras que se plantean, y las investigaciones y adiciones que se pudiesen efectuar a continuación del proyecto.

2. Estado del arte

2.1. Introducción

Se hará en este capítulo un análisis y revisión en líneas generales de todas las tecnologías que aparecen en el documento, con idea de darle al lector unos conocimientos generales del marco en el que se desarrolla el trabajo.

2.2. Sistema IoT

Una definición de un sistema IoT es un conjunto e interconexión de una serie de objetos a lo largo de una red (ya sea privada o la propia Internet) y dispositivos. Se ha de tener en cuenta que cualquier cosa que uno se plantee es posible modelizarla, conectarla e incluso interaccionar con ella a través de una red. Siendo el objetivo principal una intercomunicación máquina a máquina (lo que se conoce en el argot como M2M) en favor a los designios humanos. En cuanto al tipo de objeto o dispositivo, cualquier sensor, dispositivo mecánico e, incluso, cualquier objeto cotidiano como un aire acondicionado, ropa o un vehículo podría serlo.

Por esta razón, se entiende la gran presencia y popularidad de los sistemas IoT hoy en día, aplicable a infinidad de situaciones como se muestra en la Figura 1, mejorando la vida cotidiana de las personas como en los entornos empresariales, en dónde su implementación lleva unos años ya. (Gracia M. , 2019)

Figura 1: La tecnología IoT presente en el día a día
Fuente: (Blac Solutions, 2015)

Según la Organización Mundial de la Salud, las enfermedades cardiovasculares representan la causa principal de muerte a nivel mundial, y alrededor de 17,9 millones de personas pierden la vida anualmente por ella. (World Health Organization, 2020)

Aplicaciones actuales, como la desarrollada por investigadores del Departamento de Ingeniería Instrumental del Instituto Tecnológico Ramrao Adik en la India (Figura 2), es un ejemplo de cómo esta tecnología aporta soluciones a problemas cotidianos. Los investigadores diseñaron un kit médico para condiciones cardiovasculares críticas, con idea de un fácil acceso a los datos de los pacientes por parte del personal de la Unidad de Cuidados Intensivos. Este conjunto de dispositivos tiene un bajo coste y mide diferentes parámetros de salud, los cuales sirven para generar un sistema de vigilancia en zonas rurales al permitir compartir los datos y poder ser evaluados por el personal médico. (Acharya & Patil, 2020)

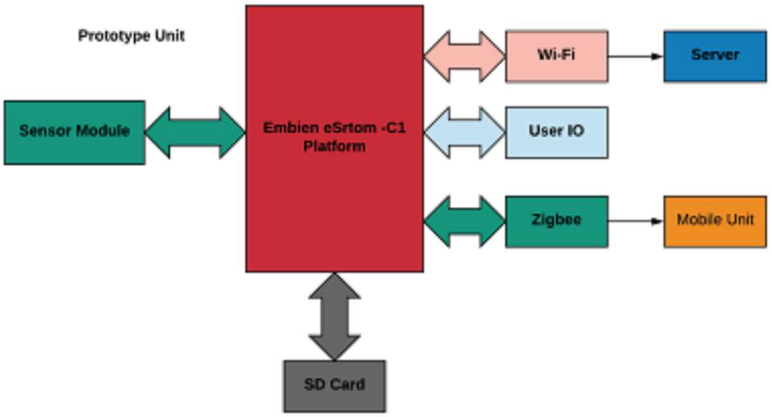


Figura 2: Prototipo de sistema de atención médica
Fuente: (Acharya & Patil, 2020)

Por otro lado, la domótica cada vez está más presente, y el hecho de poder montar un sistema a bajo coste y que permita desde un simple smartphone controlar y monitorizar cada uno de los elementos implicados en él. Un ejemplo de esto es el desarrollado por estudiantes de la Facultad de Ingeniería Eléctrica de Terengganu, en Malasia. Los estudiantes han generado una solución para controlar una serie de elementos en las habitaciones de su residencia de estudiantes. A través de sus teléfonos móviles mediante una aplicación, son capaces de encender la luz de la habitación, el ventilador; o accionar una puerta corredera e incluso una cortina. En la figura 3 se muestra el diagrama de bloques que describe el sistema. (Mahzan, 2020)

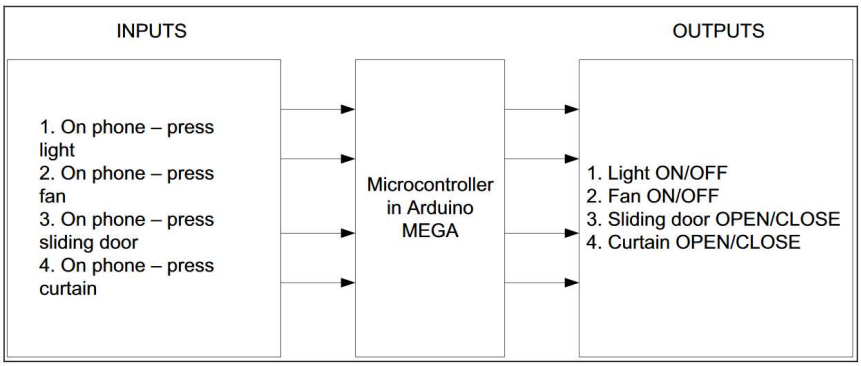


Figura 3: Diagrama de bloques de la habitación inteligente
Fuente: (Mahzan, 2020)

2.3. Transmisiones inalámbricas

Cada tipo de aplicación tiene una serie de requisitos específicos que se han de tener en cuenta a la hora de elegir correctamente que tipo de tecnología de comunicación se adapta mejor al problema a resolver. Singularmente, las tecnologías inalámbricas relacionadas con IoT desarrolladas en los últimos años son totalmente heterogéneas en cuanto a protocolos, rendimiento y cobertura. Es decir, algunas de ellas están concebidas directamente para las radiocomunicaciones de corto alcance (por ejemplo, Bluetooth y ZigBee), otras son más adecuadas para cubrir zonas amplias con un ancho de banda muy pequeño (por ejemplo, Sub-GHz), mientras que otras están diseñadas para comunicaciones de rango medio y alta tasa de transmisión (por ejemplo, IEEE 802.11) (Cilfone, Davoli, Belli, & Ferrari, 2019). A continuación, se van a revisar los diferentes protocolos, presentando sus principales características para su uso en un sistema IoT.

2.3.1. Wi-Fi IEEE 802.11

El estándar 802.11 define las propiedades de una red de área local inalámbrica (WLAN). En cuanto a Wi-Fi (*Wireless Fidelity*) es una certificación otorgada por la Wi-Fi Alliance (Wi-Fi Alliance, 2020), el cual garantiza que los dispositivos que utilizan el estándar 802.11 son compatibles entre sí.

Así pues, con esta tecnología se pueden crear redes inalámbricas de área local con una velocidad alta de transmisión de datos, con la salvedad de que el equipo a conectar esté próximo al punto de acceso, esto permite la conexión de cualquier dispositivo, que soporte una conexión de velocidades superiores a los 11 Mbps, y que se encuentre en un radio entre 20 y 100 metros dependiendo de si es un entorno cerrado o abierto.

Se ha de destacar que el estándar 802.11 es, en realidad, el primer estándar y tan sólo ofrece un ancho de banda de 1 a 2 Mbps, con lo que se ha modificado con idea de optimizar el ancho de banda y garantizar en algunos casos mayor seguridad o compatibilidad. En la Tabla 1 se muestra una equivalencia de tres

modificaciones que se han realizado al 802.11, los cuales operan de modos diferentes, permitiéndoles llegar a diferentes velocidades y rango, lo que abre un abanico de posibilidades en función de su aplicación. (Villagómez, 2018)

Tabla 1: Comparativa rango y flujo de datos de 3 modificaciones del estándar 802.11
Fuente: (Villagómez, 2018)

Estándar	Frecuencia	Velocidad	Rango
wifi a (802.11a)	5 GHz	54 Mbit/s	10 m
wifi B (802.11b)	2,4 GHz	11 Mbit/s	100 m
wifi G (802.11b)	2,4 GHz	54 Mbit/s	100 m

2.3.2. Bluetooth Mesh

Es una especificación de red de malla basada en BLE (*Bluetooth Low Energy*) (Bluetooth, 2020), utilizada en escenarios en donde son necesarios la comunicación inalámbrica de baja energía. La red mallada permite una topología en la que cada nodo de la red tiene la posibilidad de comunicarse con los otros nodos que la conforman.

Su funcionamiento es similar al de un sistema de mensajería, donde los mensajes se distribuyen según el principio de *publishing* (envío de mensaje a dirección específica) *and subscribing* (configuración del dispositivo a la que se le otorga una dirección específica). De tal forma, que el dispositivo tan sólo puede recibir mensajes si se le envía a la dirección específica. Así pues, el funcionamiento del Bluetooth Mesh consiste en que cada mensaje se envía a todos los dispositivos al alcance, hasta llegar al nodo correcto. La ventaja de este sistema está en que, si el dispositivo que envío el mensaje estaba fuera de rango, el mensaje puede llegar al nodo correcto si el número de participantes en la red es elevado. (Redeweb, 2019)

La Figura 4 muestra algunas funciones opcionales que pueden implementarse en los nodos con idea de gestionar y mejorar la comunicación.

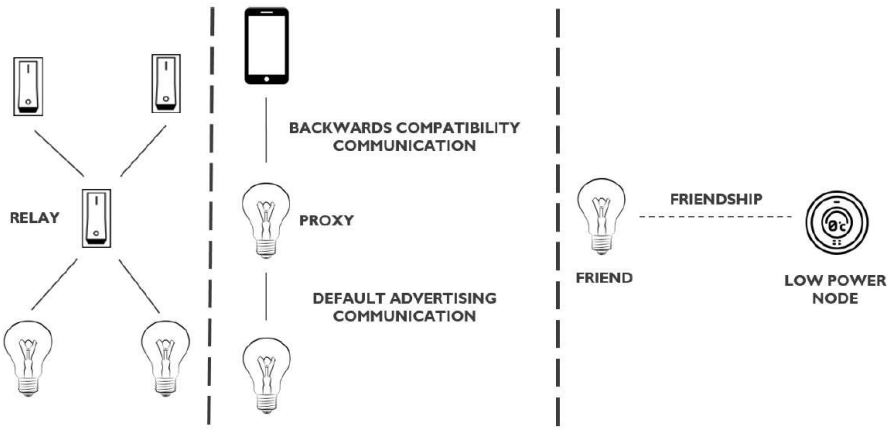


Figura 4: Ejemplo de cada una de las funciones del estándar Bluetooth Mesh.
Fuente: (Baert, Rossey, Shahid, & Hoebeke, 2018)

Por ejemplo, en el caso de usar el nodo como relé, hará que todo mensaje recibido sea reenviado a la red. También se incluye un caché de mensajes para asegurar que el nodo retransmite dicho mensaje una única vez. Esta función se observa en la parte izquierda de la Figura 4.

En cuanto a la parte central de la Figura 4 se muestra como el nodo actúa como proxy. De forma que al conectar un smartphone que es un dispositivo BLE nativo, podría comunicarse con la bombilla a través de una bombilla intermedia que actuaría como proxy, reenviando el mensaje a la red y actuando en este caso también como un relé.

Finalmente, en la parte derecha de la Figura 4 se observa como funciona la característica de amistad. Al estar constantemente los nodos trabajando al 100% escaneando los diferentes canales para publicar, entra el conflicto el sentido de nodos que usen BLE, y limita el sentido de aplicaciones de baja energía. Por esta razón se usa la función de amistad, en donde los nodos de la red mallada que estén conectados a la red eléctrica actúan como “amigos” de aquellos nodos de baja potencia, almacenando los mensajes entrantes destinado al nodo de baja potencia y reenviando los mensajes a la red. De tal forma que el nodo de baja potencia le pide nuevos mensajes a su “amigo” en ciertos intervalos de tiempo. Permitiendo que pueda ahorrar energía al no necesitar un ciclo de trabajo del 100%. (Baert, Rossey, Shahid, & Hoebeke, 2018)

2.3.3. ZigBee IEEE 802.15.4

ZigBee (ZigBee Alliance, 2020) es un protocolo inalámbrico basado en IEEE 802.15.4, siendo este el estándar más adecuado en cuanto a operar en condiciones climáticas o ambientales adversas. Su propósito principal es el de ser una solución para aquellas aplicaciones en donde sean necesarias comunicaciones seguras, maximizar la vida de las baterías y en las que se precisen una tasa de envío de datos baja. Con las cualidades citadas anteriormente, se saca la conclusión de que la domótica es un ámbito perfecto para aplicar esta tecnología, más aún cuando otras características de este protocolo son la posibilidad de realizar una red mallada y una integración sencilla con bajo coste.

Según se configuren, los dispositivos ZigBee, podrán realizar las siguientes funciones:

- Coordinador: en esta configuración el dispositivo puede comunicarse con todos los demás nodos de la red y puede controlar incluso todos los dispositivos de la red que estén conectados al coordinador. Dentro de una red, tan sólo puede existir un coordinador.

- Router: en este modo tiene la capacidad de registrar y transmitir los datos de los sensores. A su vez, también puede comunicarse con el coordinador y los dispositivos finales. Aunque su función principal es la de aumentar si fuera necesario, intensidad de la señal de los paquetes de datos en caso de que esta fuera débil. No existe limitación a la cantidad de routers a desplegar en una red.

- Dispositivo final: en esta modalidad puede comunicarse solamente con un router, pero no con el coordinador[1]. El dispositivo final transmite los datos a un router y este a su vez al coordinador o a otro router en función de la

[1] Versiones nuevas del firmware permiten la comunicación con el coordinador también

distancia. Al igual que los routers, pueden existir en la red tantos como sean necesarios. (Sahitya, Balaji, Naidu, & Abinaya, 2017)

Según Dávila, Pérez, Mantilla, y Moreno (2016) con los dispositivos definidos anteriormente, se pueden realizar tres tipos de topologías de redes diferentes: estrellada, mallada y de tipo árbol. En la Figura 5 se muestra cada una de las estructuras de redes posibles.

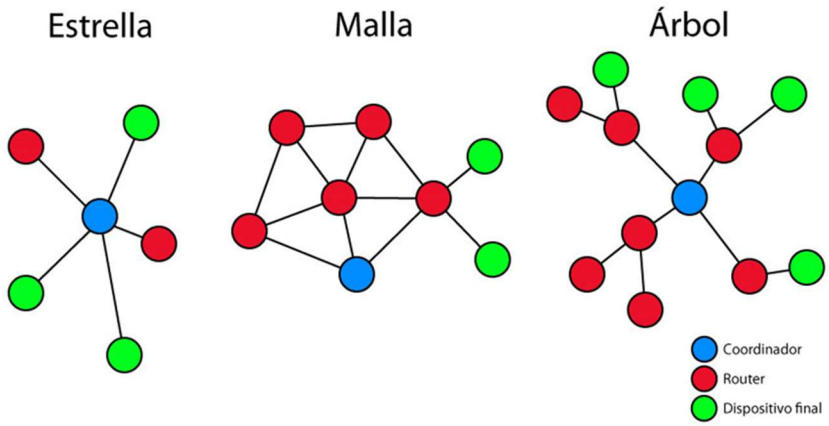


Figura 5: Tipología de red ZigBee
Fuente: Elaboración propia

- Red estrellada: en donde el nodo coordinador se encarga del enlace de todos los dispositivos existentes en la red.

- Red mallada: existe un coordinador, pero las conexiones se pueden realizar entre dispositivos que estén disponibles en el mismo rango y a su vez, es capaz de adaptarse para ampliar el alcance de la red y tener enlaces redundantes.

- Red de tipo árbol: red ramificada en dónde el coordinador ocupa una posición centralizada de la que parten diversos brazos conformados por otros dispositivos en modo router y/o dispositivos finales.

2.3.4. LPWAN

Como se ha visto hasta ahora, cada uno de los estándares permiten la comunicación inalámbrica entre los dispositivos IoT, y a su vez, cada uno de ellos mediante puertas de enlace (*gateways*) con Internet.

Se ha expuesto también que cada uno tienen diferentes especificaciones, no obstante, se podrían dividir en dos grandes categorías: corto alcance (menos de 100 metros de radio) y largo alcance (mayor a 1000 metros de radio). Dentro de la primera categoría estarían incluidos los tres estándares vistos con anterioridad: IEEE 802.15.4, Bluetooth y ZigBee. Y en la segunda categoría estarían aquellos protocolos similares a las redes celulares o redes móviles. (Fialho & Azevedo, 2018)

En el caso del estándar LPWAN (*Low power wide area network*), de sus siglas se entiende que está destinado para la conectividad de dispositivos que requieran un ancho de banda inferior al habitual. La finalidad de este tipo de redes es la de transferir datos a baja velocidad con idea de mantener un bajo consumo energético y de esta forma prolongar la vida útil de las baterías. (Osiberia, 2016)

En cuanto a la gran variedad de tecnologías dentro de este estándar, hay que destacar dos de ellas que están ganando una popularidad mayor que otras LPWAN: SigFox (SIgFox, 2020) y LoRa (*Long Range modulation*) (LoRa Alliance, 2020). Ambos protocolos operan también a través de las bandas de radio industriales, ciéntifcas y médicas (ISM), siendo estas una parte específica del espectro de radio reservada internacionalmente para los usos anteriormente descritos.

La diferencia principal entre ambas es que mientras que el estándar SigFox limita la subida y descarga de mensajes por día, siendo necesaria una suscripción para su uso, LoRa es un protocolo abierto, realizado por la asociación sin ánimo de lucro LoRa Alliance que permite la creación de una red privada sin limitación de tráfico de datos, tamaño de paquetes y sin suscripción. (Fialho & Azevedo, 2018)

2.3.5. Comparativas tecnologías de transmisión inalámbricas

Tras el análisis realizado en los apartados anteriores, se saca una serie de conclusiones que permiten discernir qué tecnología de transmisión inalámbrica es la más adecuada en función de las casuísticas del sistema que se pretenda realizar. En la Figura 6 se muestra una gráfica comparativa de todos los estándares vistos anteriormente en base a la velocidad de transmisión de datos y el alcance.

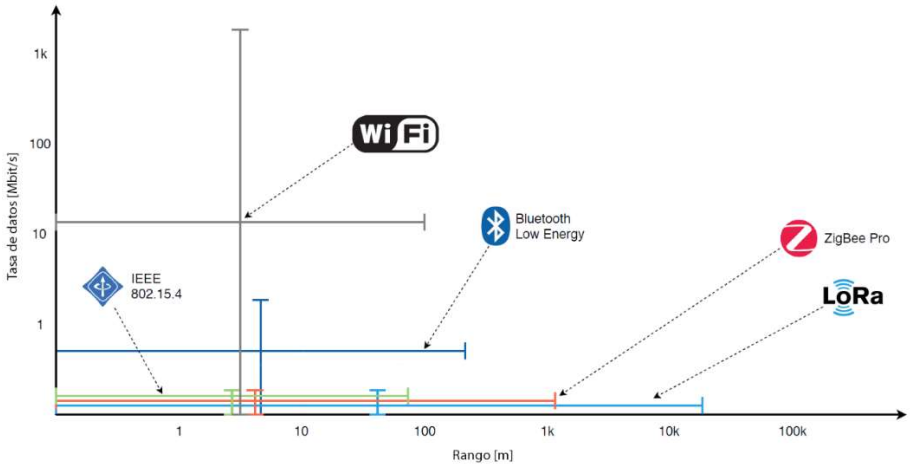


Figura 6: Comparativa entre los protocolos de comunicación
Fuente: (Cilfone, Davoli, Belli, & Ferrari, 2019) y editada

Por ejemplo, en una aplicación en donde es necesario una baja tasa de datos, un rango de alcance elevado y un consumo energético bajo; el mejor estándar sería LPWAN. Ya dentro de este estándar se elegiría LoRa o SigFox dependiendo del coste económico del proyecto.

En cambio, como alternativa ZigBee podría servir como una tecnología de malla adecuada, especialmente en términos de escalabilidad y consumo energético. Y si se precisara de un alcance superior, el uso de ZigBee Pro sería una solución más que idónea. No obstante, si se necesitara un rango más corto, Bluetooth Mesh mediante BLE podría ser una opción interesante, aumentando así la tasa de datos y, proporcionando un buen rendimiento en término de topología de red, y cobertura (aunque requiere de un número elevado de nodos conectados). Por

último, el uso de del protocolo IEEE 802.11 garantizaría una transmisión aceptable, con el rango más alto de velocidad de todos los nombrados anteriormente, pero en defecto con un consumo energético muy superior.

Con objetivo de tener una visión más completa, en la Figura 7 se presenta una comparativa de las tecnologías inalámbricas más aún en detalle. Se tomaron en consideración los siguientes parámetros:

- Cobertura (m^2): se entiende como la superficie que puede cubrirse mediante una red de malla según la tecnología analizada.
- Rango (m): siendo el alcance de transmisión de un único nodo en la red mallada.
- Escalabilidad: la capacidad de aumentar la red existente.
- Tasa de datos (b/s): Velocidad de la transmisión de datos medido en un nodo.
- Topología de red: se entiende como el grado de complejidad alcanzable al construir cada tipo de red.
- Vida de la batería (días): medido en los nodos individualmente.
- Consumo eléctrico (W): medido en los nodos individualmente.
- Latencia (s): se concibe como la capacidad de cada estándar en cuanto a baja latencia durante las comunicaciones entre nodos.
- Despliegue: se entiende como la complejidad que surge al desplegar una red de malla según tecnología de transmisión. (Cilfone, Davoli, Belli, & Ferrari, 2019)

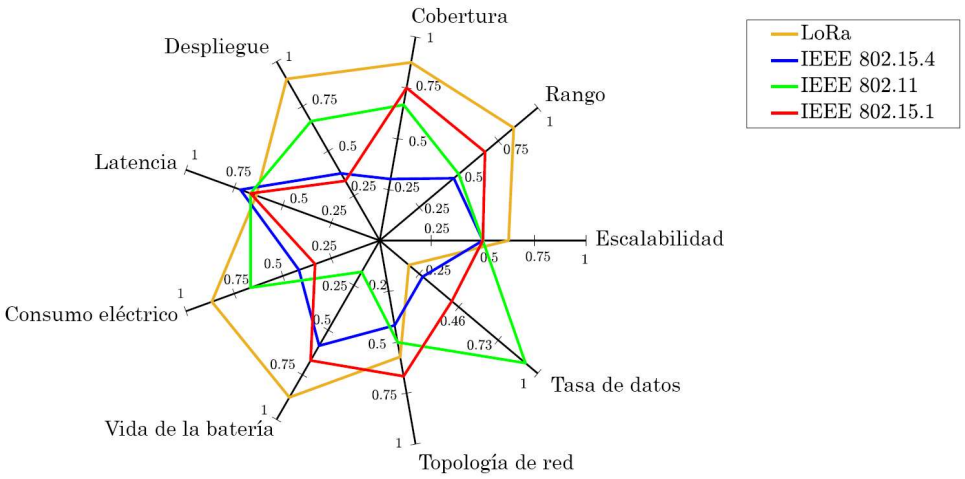


Figura 7: Análisis comparativo de prestaciones
Fuente: (Cilfone, Davoli, Belli, & Ferrari, 2019) y editada

2.4. Repositorios en la nube

Una vez finalizado un proyecto de IoT surge la duda de cuál es la mejor manera de monitorizar los datos que se captan por los sensores. Partiendo de la premisa más básica, se encuentra la opción de usar cualquier *display* como un LCD o una pantalla TFT e incluso un monitor si existe la posibilidad de conexión.

No obstante, existen en la actualidad una gran cantidad de plataformas web que permiten la visualización de los datos e incluso algunas tienen la capacidad de mostrar un histórico de los mismos, pudiendo una vez almacenada la información, aplicarle varias técnicas de predicción y de gestión. (del Valle, Programar facil, 2020)

La subida de datos hacia estas plataformas se efectúa utilizando un protocolo específico de comunicación, lo más comunes son:

- HTTP (*Hypertext Transfer Protocol*): permite realizar una petición de datos y recursos, por ejemplo, un documento. Para ello, cliente y servidor se comunican intercambiando mensajes individuales. Llamados petición los enviados por el cliente, normalmente un navegador Web, y los enviados por el servidor llamados respuestas. (MDN, 2020)

- MQTT (*Message Queue Telemetry Transport*): protocolo de comunicación M2M. Se basa en la pila TCP/IP[2] como base de su comunicación a diferencia del sistema HTTP. Es decir, en este caso cada conexión se mantiene abierta y se reutiliza en cada comunicación. (Llamas, Luis Llamas, 2019)

- CoAP (*Constrained Application Protocol*): implementa el modelo REST[3] de HTTP, e incluye además *multicast*, es decir, permite el envío de la información a múltiples destinos simultáneamente desde múltiples redes. (Gracia L. , 2013)

En cuanto a qué plataforma escoger, dependerá de la magnitud del sistema IoT, y del coste a soportar por su uso. Un primer grupo sería el de aquellas plataformas orientadas a *startups* y empresas pequeñas. Permiten un uso gratuito, pero con limitaciones al número de mensajes que se pueden enviar y de dispositivos a conectar. Es el caso de aREST Framework (aREST, 2020), como su nombre indica es un entorno de trabajo (*framework*) que utilizar el protocolo REST. Ofrece un sistema gratuito con limitaciones y distintos planes de suscripción temporal (Tabla 2). Su gran ventaja es su sencillez y que permite su descarga para proyectos en local, y por contrapartida la documentación es bastante escasa.

Tabla 2: Planes de suscripción aREST Framework
Fuente: Elaboración propia

	Starter	**Hobbyst**	**Developer**
Precio	Gratis	9€/mes	19/mes
Nº dispositivos	5	25	100
Nº mensajes	100	10.000	100.000
Soporte Email	Estándar	Prioritario	Prioritario

[2] Es un conjunto de protocolos interconectados que define un método para comunicar dispositivos a través de una red.
[3] Tipo de arquitectura de desarrollo web que usa un modelo de solicitud para la tramitación de datos. Las operaciones más comunes son GET, POST, PUT, PATCH y DELETE.

Más interesante si cabe es ThingSpeak (ThingSpeak, 2020), llevada a cabo por MathWorks (MathWorks, 2020), los creadores de MATLAB y *Simulink*. También dispone de un método gratuito con sus limitaciones y de pago (Tabla 3). Las posibilidades que ofrece este servicio son la de subida, visualización, análisis de datos en vivo en la nube y envió de alertas. Su gran potencial es que dispone de diferentes *Apps*[4], desde el uso de una aplicación de MATLAB para visualizar y analizar los datos, hasta un sistema para publicar *tweets* en *Twitter*. Otra gran ventaja es la gran comunidad de usuarios de la web, los cuales participan activamente en el foro, resolviendo dudas y mostrando sus proyectos.

Tabla 3: Planes de suscripción ThingSpeak
Fuente: Elaboración propia

	Gratuita	**Estándar**
Nº Mensajes	3.000.000/año	33.000.000/año
Frecuencia subida datos	Cada 15 seg	Cada seg
Nº Canales	4	250

Por otro lado, estarían las plataformas ofrecidas por grandes empresas como Microsoft, IBM o Google. En donde no sólo se contempla la tramitación de datos, si no gran capacidad de almacenamiento, API[5] para dispositivos móviles y bases de datos. Especialmente diseñadas para uso profesional, gestión de maquinaria y sector industrial; en definitiva, en cualquier área donde se requiera un control de máquinas, dispositivos u objetos de gran escala. Ejemplo de ellas son: IBM Watson IoT (IBM, 2020) y Google Cloud Platform IoT (Google, 2020)

Finalmente, las plataformas de código abierto como Zetta (Zetta, 2020) o Node-RED (Node-RED, 2020), en las que se tiene el acceso al código sin restricciones, permitiendo descargarlo e instalarlo para su uso en un dispositivo de forma local. No se recomienda su uso para usuarios ajenos a tecnología de servidor, ya que suelen estar basados en servidores web como Apache o NodeJS. (del Valle, Programar facil, 2020)

[4] Programa o aplicación informática generalmente pequeña que cumple una función específica.
[5] Conjuto de procesos, funciones y métodos derivados de una determinada biblioteca de programación para ser empleada por otro programa informático.

2.5. Sistemas microcontrolador y ordenadores de placa única

A la hora de abordar un proyecto de IoT es importante estudiar el alcance de este, con el objetivo de estructurar cada pieza del sistema para que encajen correctamente, evitando costos innecesarios, duplicidad de funciones entre dispositivos e incluso evitar 'matar moscas a cañonazos', como sería el uso de elementos o aparatos con capacidades muy por encima de las necesarias.

En secciones anteriores se han revisado los principales protocolos de transmisión inalámbrica, y repositorios en la nube. En este apartado se van a revisar una serie de dispositivos de hardware comerciales, dividiéndose en dos grandes grupos: microcontroladores y ordenadores de placa única.

Un microcontrolador es básicamente un circuito integrado en cuyo interior alberga una CPU (unidad central de procesamiento), unidades de memoria (RAM y ROM), puertos de entrada y salida y periféricos. El propósito de estos microcontroladores es el de leer y ejecutar los programas escritos por el usuario y almacenarlos en su memoria. (Electronica Estudio, 2020). En la Tabla 4, se muestra una comparación de 3 microcontroladores usados en proyectos IoT: Arduino Uno (Arduino, 2020), Intel Galileo (Intel, 2020) y LPC1768 (NXP, 2020).

Tabla 4: Comparación microcontroladores
Fuente: Elaboración propia

	Arduino Uno Rev3	**LPC1768**	**Intel Galileo**
Voltaje operativo (V)	5	4,5 - 9	3,3
Procesador	ATmega328P	ARM Cortex-M3	Intel Quark SoC X1000
Compatibilidad software	IDE	Basado en la nube y C/C++	IDE, Eclipse y Python
SRAM (KB)	2	64 (32+32)	512
EEPROM (KB)	1	512	11 (permite introducir SD de hasta 32GB)
Pines digitales I/O	14 (6 PWN)	20	14 (6 PWN)
Pines analógicos I/O	6	8	8
Puertos series	UART, I2C, SPI, USB	UART, I2C, SPI, USB	UART, I2C, SPI, USB
WiFi	Sí	Sí	Sí
Consumo medio (mA)	50	200	101,3

Dimensiones (mm)	68,6 x 53,4	54 x 26	107 x 74
Precio aproximado	20 €	45 €	60 €

Como conclusión de la tabla anterior se obtiene que:

- En un proyecto en donde se necesite una capacidad de computación y memoria mayor, la mejor opción es el microcontrolador Intel Galileo.
- En donde el consumo sea una limitación, la mejor opción es Arduino Uno Rev3.
- En cuanto a Pines digitales, analógicos y puertos series; ambos tres son prácticamente iguales.
- Si el precio es determinante, Arduino Uno Rev3 es más competente de los tres.

Por otro lado, se encuentran los ordenadores de placa única (SBC). La gran diferencia de estas con respecto a las anteriores es que traen de base una capacidad de computación superior y de almacenamiento (Güven, Cosgun, & Kocaoglu, 2017). Se enumeran a continuación, 3 modelos populares de SBC: Raspberry Pi 3 Model B (Raspberry Pi, 2020), Le Potato (Libre Computer, 2020) y Odroid-Xu4 (Hard Kernel, 2020).

Tabla 5: Comparación SBC
Fuente: Elaboración propia

	Raspberry Pi 3 Model B	**Le Potato**	**Odroid-Xu4**
Voltaje operativo (V)	3,3	5	5
Procesador	Broadcom BCM2837, Cortex-A53 (ARM) 64-bit 1.2GHz	Quad 64-bit 1,5 GHz ARM Cortex-A53, 4 K Ultra HD ARM Mali 450 GPU de 750 MHz	Samsung Exynos 5422 Cortex-A15 2Ghz y Cortex-A7 Octa core
RAM (GB)	1	2	2
Memoria (GB)	SD 8 a 64 GB	SD 8 a 64 GB	SD 8 a 64 GB
Pines GPIO	40	40	42
ADC[6]	No	No	Sí

[6] ADC es un convertido de señal analógica a digital.

Puertos series	UART, I2C, SPI, USB	UART, I2C, SPI, USB	UART, I2C, SPI, USB, ADC
WiFi	Sí	No	No
Ethernet	Sí	Sí	Sí
Bluetooth	Sí	No	No
Consumo medio (A)	2,5	2,5	4
Dimensiones (mm)	82 x 56	122 x 77	83 x 58
Precio aproximado	34 €	40 €	67 €

Frente a la evidencia recaudada en tabla anterior se puede afirmar que:

- Los SBC tienen una capacidad de computación y memoria superiores a la de los microcontroladores. Y de los tres, el más potente es Odroid-Xu4.
- Aunque Odroid-Xu4 sea el único con ADC, la diferencia de precio existente con la Raspberry Pi 3 Model B hace que esta sea más interesante, puesto que añadirle un Shield con ADC sigue siendo más económico.
- El hecho de que Raspberry Pi 3 Model B disponga de WiFi y Bluetooth de forma nativa, lo hace ser una elección casi segura en la mayoría de los proyectos de IoT.
- En cuanto a consumo, tanto Raspberry Pi 3 Model B y Le Potato son una opción más que interesante.

2.5.1. Periféricos

El propósito principal de este apartado es saber qué es un periférico, la tipología y sus usos.

Según Raspberry (2020), una de las características más potente que tiene la Raspberry Pi es su hilera de pines GPIO (pines de entrada o salida de propósito general). Cualquiera de estos pines puede ser designado mediante software como una entrada o salida y a su vez para una gran variedad de funciones.

Los pines configurados como entrada y salida pueden realizar las siguientes funciones:

- Alimentación: de base la Raspberry Pi dispone de dos pines a 5V y otros dos a 3,3V.

- Entrada: en esta configuración el pin puede leer una tensión de 3,3V como un High o 0V como un Low, definiendo de forma binaria un valor de 1 o 0 respectivamente.

- Salida: da un valor de High o Low como salida del pin. Además, puede ser configurado como una salida de Pulso Con Modulación (PWM) permitiendo emular una señal analógica de tensión. (Llamas, 2015)

Según Aprendiendo Arduino (2017) en cuanto a su configuración como puerto serie se distinguen:

- Inter-Circuitos Integrados (I2C): es un bus de comunicaciones que usa dos líneas para la transmisión de la información, una para los datos y otra para la señal del reloj.

- Recepción-Transmisión Asíncrona Universal (UART): a diferencia del I2C este protocolo serie es asíncrono. Para transmitir utiliza una línea de datos simples y otra línea para recibir datos, de forma general 8 bits de datos son transmitidos de la siguiente manera: un bit de inicio a nivel bajo, 8 bits de datos y un bit de parada a nivel de alto.

- La Interfaz Periférica Serial (SPI): Al igual que I2C es un protocolo síncrono, en donde un maestro envía la señal del reloj, y en cada pulso envía un bit al esclavo y a su vez recibe un bit de este.

- Puerto Serial Universal (USB): reservado para entrada y salida de dispositivos informáticos como, por ejemplo, ratón, teclado o cámaras.

3. Tecnologías empleadas

En este apartado se van a mostrar cada una de las principales herramientas que han sido utilizadas en el desarrollo del proyecto, y se dividirán en dos grandes grupos: hardware y software.

3.1. Hardware

3.1.1. Raspberry Pi 3 Model B

La Raspberry Pi 3 Model B (Figura 8) es un ordenador de placa única (SBC) del tamaño de una tarjeta de crédito y de bajo coste. Al disponer de una cantidad notable de pines GPIO y puertos series, se entiende su popularidad en proyectos de IoT. Debido a este uso extendido, existe una gran cantidad de documentación tanto en medios digitales como físicos.

Figura 8: Raspberry Pi 3 Model B
Fuente: (Raspberry Pi, 2020)

En la Tabla 6 se muestran sus especificaciones técnicas:

Tabla 6: Especificaciones Raspberry Pi 3 Model B
Fuente: Elaboración propia

Raspberry Pi 3 Model B	
Procesador	Quad Core 1.2GHz Broadcom BCM2837 64bit
Ram	1GB
Memoria	512 MB
Almacenamiento	Tarjeta SD desde 8 GB hasta 64 GB
Puertos USB 2.0	4
Salidas de vídeo	HDMI
	Puerto CSI para cámara
	Puerto DSI para pantalla LCD
Salidas de audio	HDMI
	Jack de 3.5 mm
Conectividad de red	RJ-45 Ethernet de 10/100 Mbps
	WiFi 802.11 b,g,n
	Bluetooth 4.1
Pines GPIO	40 (incluye UART, I2C, SPI)
Sistemas operativos soportados	GNU/Linux: Raspbian
	Fedora (Pidora)
	Arch Linux
	Slackware Linux

Para su aclaración, en la Figura 9 se muestra la distribución de los pines GPIO.

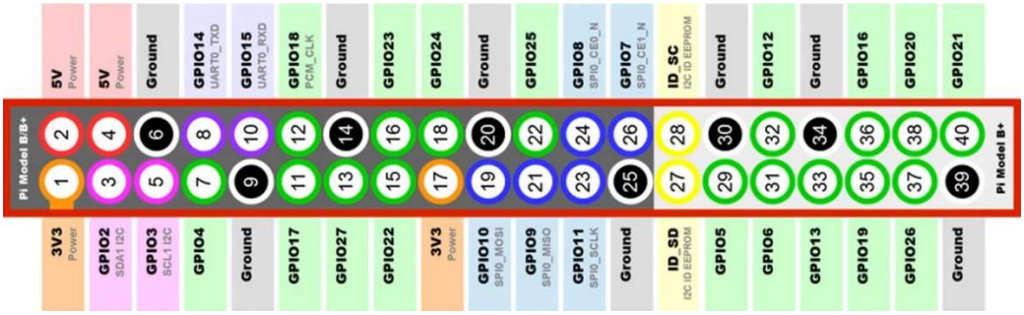


Figura 9: Distribución pines GPIO
Fuente: (Rogelio, 2015)

Se ha realizado la Figura 10 con el objetivo de esclarecer cualquier duda sobre la ubicación de cada una de las especificaciones enunciadas en la Tabla 6.

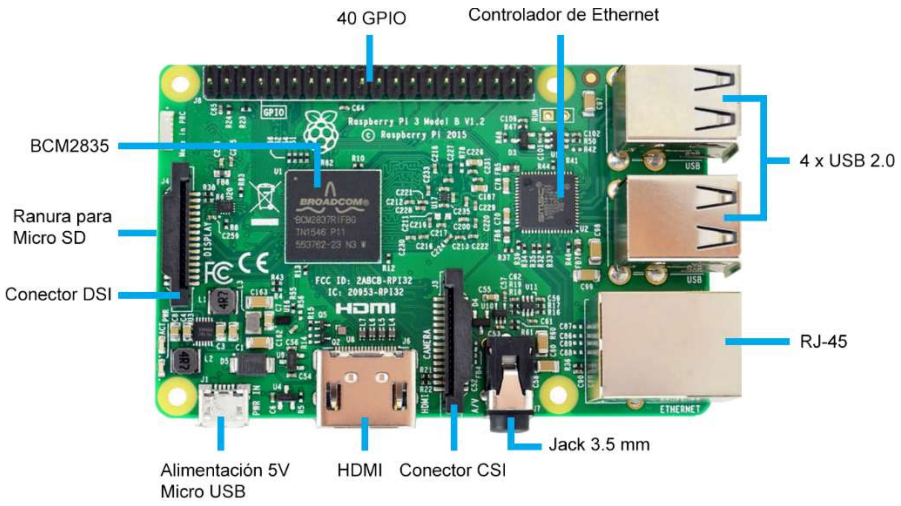


Figura 10: Ubicación de componentes Raspberry Pi 3 Model B
Fuente: Elaboración propia

Por último, la gran capacidad de procesamiento y memoria frente a un microcontrolador, además del bajo coste y que incorpora WiFi de serie, fueron las razones por las que las Raspberry Pi 3 Model B fueron elegidas para el proyecto.

3.1.2. Raspberry Pi to Arduino Shield

Uno de los principales inconvenientes que tiene la Rasberry Pi es la ausencia de pines analógicos, impidiendo que se puedan utilizar de base sensores analógicos. Para resolver este inconveniente se puede hacer uso de un 'shield', es una placa de circuito modular que se acopla encima de un dispositivo con el objetivo de añadir la funcionalidad extra deseada.

En este proyecto se utilizó el shield Raspberry Pi to Arduino Connection Bridge Version 2 de la compañía Coocking Hacks (Coocking Hacks, 2020), división comercial de Libelium (Libelium, 2020). Una vez conectado el shield a la Raspberry Pi posibilita el uso de hasta 7 pines analógicos y la conexión del

módulo XBee en un socket[7] específico, además permite que se conecten a él cualquier otro shield, placa o módulo compatible con Arduino.

En la figura 11 se muestra en detalle el shield:

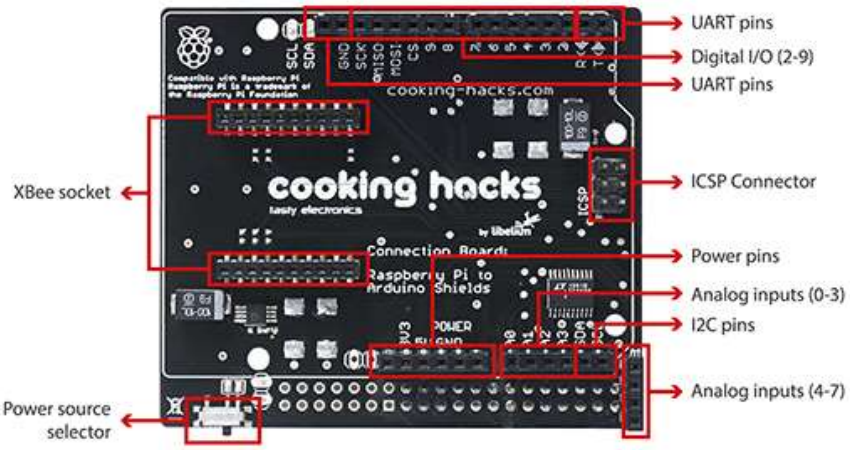


Figura 11: Raspberry Pi to Arduino Connection Bridge Version 2
Fuente: (Coocking Hacks, 2020)

Finalmente, se adjunta en el Anexo III el diagrama esquemático del shield.

3.1.3. XBee Pro S1

De acuerdo con Digi (2019): "los módulos XBee son soluciones integradas que brindan un medio inalámbrico para la interconexión y comunicación entre dispositivos. Estos módulos utilizan el protocolo de red llamado IEEE 802.15.4 para crear redes POINT-TO-MULTIPOINT (punto a multipunto); o para redes PEER-TO-PEER (punto a punto). Fueron diseñados para aplicaciones que requieren de un alto tráfico de datos, baja latencia y una sincronización de comunicación predecible. Por lo que básicamente XBee es una implementación de Digi basada en el protocolo Zigbee. En términos simples, los XBee son módulos inalámbricos fáciles de usar."

En cuanto a las comunicaciones ZigBee se realizan en la banda libre de 2.4GHz, en un único canal sin salto, a diferencia de Bluetooth que utiliza un salto de

[7] Socket que permitirá la conexión con el puerto serie.

frecuencia para minimizar los efectos de interferencia. En teoría, el número máximo de dispositivos que pueden formar una red asciende a 65.535. Al mismo tiempo, la velocidad de transmisión de datos es de hasta 256 kbps y el alcance dependerá de la potencia de transmisión de Xbee (puede ser configurada) y de la antena utilizada (Oyarce, 2008). En el caso del módulo Xbee (Figura 12) usado en el proyecto es la versión Pro, el alcance en interiores comprende los 100 m y en línea vista hasta los 1.600 m.

Figura 12: XBee Pro S1 y antena RP-SMA 2.4 GHz

Finalmente, el bajo coste y consumo de potencia, junto con la capacidad de usar bandas de radio libres sin necesidad de licencias y su fácil instalación; fueron las razones por las que el módulo XBee Pro S1 fue escogido como solución de comunicación inalámbrica para el proyecto.

3.1.4. Sensores

Conforme a Ruiz (2019), un sensor es un dispositivo que tiene la capacidad de captar estímulos del entorno y a su vez mediante un transductor convertirlos en un impulso eléctrico, que a su vez será analizado, procesado y transformado, con el objetivo de generar una determinada respuesta. En cuanto a la información del exterior puede ser de distinta naturaleza (mecánica, química, acústica…).

Existe una gran variedad de sensores según el tipo de variable que deban medir. Por lo tanto, se van a enumerar tan sólo aquellos que han intervenido en el proyecto:

- LM35DT: según Hetpro (2017), es un sensor de temperatura analógico[8], es decir, suministra un voltaje proporcional a la temperatura, esto es, 10mV por cada 1°C. Tal como sugiere la siguiente fórmula:

$$v_{out} = 10\frac{mV}{°C} \cdot T$$

Donde:

v_{out} es la tensión de salida del sensor.

T es la temperatura en ºC.

El sensor opera entre 4 y 30V de alimentación y tiene una precisión de 0.5 ºC, en la Figura 13 se muestra el sensor utilizado.

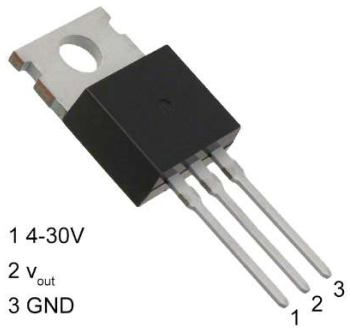


Figura 13: Sensor de temperatura LM35DT con encapsulado TO-220

- HC-SR04: conforme a Llamas (2015), es un sensor de ultrasonido que sirve para medir distancias, su rango[9] de medición oscila entre 2 a 400 cm con una tolerancia de 0.3 cm. Se basa en la medición entre el envío de un pulso sonoro desde el pin Trigger y su recepción desde el pin Echo, es decir, su modelización matemática sería:

[8] el ADC del shield tiene una resolución de 10 bits lo que implica que la salida toma valores desde 0 a 1.023, es decir, (2^{10}-1). Esto se tendrá en cuenta a la hora de realizar el código de lectura del sensor.

[9] Al ser un sensor de baja precisión y poco adecuado en entornos con varios objetos, su rango real está en torno a 20 cm hasta 200 cm.

$$D = V \cdot \frac{T}{2}$$

Siendo:

D la distancia recorrida en cm.

V la velocidad del sonido en cm/µs

T el tiempo total de ida y vuelta en µs

El sensor trabaja a 5V, en la Figura 14 se muestra el sensor HC-SR04.

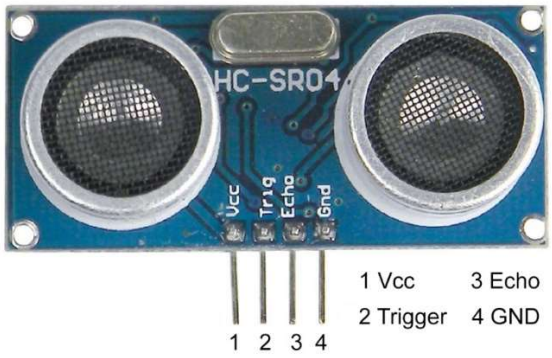


Figura 14: Sensor de ultrasonido HC-SR04

- DHT11: citando a del Valle (2017), es un sensor que permite medir la temperatura y humedad relativa (Figura 15). Realmente es un dispositivo analógico, pero al incorporar una PCB[10] con un ADC, permite su conexión a un pin digital.

[10] Placa de Circuito Impreso.

Figura 15: Sensor DHT11 con PCB

Partiendo de una señal analógica que tras su paso por el ADC integrado se transforma en formato digital, y a su vez es enviado al SBC. La información se transmite en una trama de datos de 40 bits que correponde al valor entero y decimal de la humedad y temperatura respectivamente, y por último se reservan 8 bit de paridad para confirmar que no haya datos corruptos, es decir, si la suma de los primeros 32 bit es igual a los 8 bits de paridad, la información se ha transmitido correctamente (del Valle, Programar fácil, 2017). En la Figura 16 se resume lo dictado anteriormente.

Figura 16: Trama de datos de humedad y temperatura DHT11

Finalmente, en la Tabla 7 se recogen las características técnicas del sensor.

Tabla 7: Características técnicas DHT11
Fuente: Elaboración propia

DHT11	
Alimentación	3,5 V a 5 V 2,5 mA
Temperatura	
Rango	0ºC a 50ºC
Precisión	a 25ºC ± 2ºC
Resolución	1ºC
Humedad	
Rango	20% RH a 90% RH
Precisión	0ºC y 50ºC ± 5% RH
Resolución	1% RH
Nota: La PCB incorpora una resistencia pull-up de 5 kΩ	

- Potenciómetro: conforme a Diaz (2016), un potenciómetro (Figura 17) es una resistencia variable, es decir, es un sistema compuesto por una resistencia fija (10kΩ en este proyecto) y un mecanismo que permite el deslizamiento de un dial conductor sobre la resistencia, modificando así su valor resistivo. Es por esta razón, que un potenciómetro tiene 3 pines, en donde los del extremo se comportan como la resistencia fija, y el central toma valores según el movimiento que se realice.

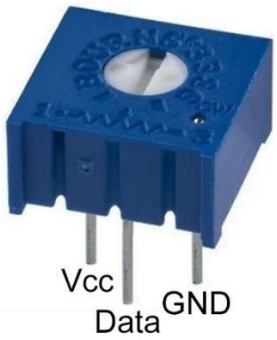


Figura 17: Potenciómetro modelo 3326P-1-103
Fuente: Elaboración propia

Para su lectura se utilizó un pin analógico del shield de Coocking Hacks, al igual que con el sensor de temperatura LM35DT devolverá un valor comprendido entre 0 y 1.023 en función de la tensión leída por el pin analógico, es decir, el potenciómetro se alimenta con una tensión de 5V la cual experimenta una caída de tensión en función del movimiento del dial.

3.2. Software

3.2.1. C++

Este lenguaje de programación fue utilizado principalmente para el desarrollo de todos los programas de lectura de sensores y transmisión de datos de la placa Raspberry Pi 3 esclava a la placa Raspberry Pi 3 maestra. La justificación del uso de C++ (Albatross, 2014) se debe a que el shield de Coocking Hacks

necesitaba hacer uso de una librería específica llamada arduPi que permite el uso del ADC.

De acuerdo con Albatross (2014), las características principales de C++ son las siguientes:

- Es de código abierto desde 1998.
- Lenguaje compilado, ganando rapidez de ejecución del programa.
- Complejo de escribir al ser de bajo nivel, pero en contra permite un gran control en lo escrito como resultado.
- Gran cantidad de bibliotecas y en crecimiento constante, simplificando la programación.
- Está orientado a objetos.
- Es compatible con casi todos los códigos en C, y puede usar librerías de C modificando un poco el código.

Finalmente, Raspbian (Raspbian, 2020) el sistema operativo de la Raspberry Pi 3 trae de forma nativa un entorno de desarrollo integrado (IDE) llamado Geany (Geany, 2020), es al fin y al cabo una aplicación informática cuyo objetivo es facilitar al programador el desarrollo del software, este IDE es específico para el desarrollo en C++.

3.2.2. Python

Si en el desarrollo del software de la Raspberry Pi esclava se utilizó el lenguaje C++, en el caso de la placa maestra fue Python (Python, 2020) en su versión 3, tanto para el desarrollo de los programas de lectura de sensores, como el del puerto serie para la adquisición de datos de la placa esclava, y el código para la comunicación con el repositorio en la nube.

En cuanto a Python, es un lenguaje interpretado, de código abierto, de alto nivel y multiplataforma. Son por estas características, la razón principal por la que este lenguaje es tan popular y de uso extendido en infinidad de proyectos de IoT. Es más, el alumno pudo comprobar al tener que hacer uso de ambos lenguajes como la curva de aprendizaje con Python fue mucho más liviana frente a la de

C++. De hecho, a la hora de buscar información en diversas fuentes, la documentación disponible en Python era más asequible desde el punto de vista de su lectura, en contra, en el caso de C++ se volvía más farragosa, debido a qué se espera que el lector tenga una base sólida en programación.

Por otra parte, el entorno de desarrollo utilizado fue Thonny (Thonny, 2020), el cual también viene instalado en Raspbian.

3.2.3. XCTU

Según Digi (2020), XCTU (Digi, 2018) es una aplicación multiplataforma gratuita que permite a los desarrolladores gestionar los módulos de radiofrecuencia (XBee en el caso del proyecto) a través de una interfaz gráfica de fácil uso. El software incluye una serie de herramientas que facilitan la configuración y localización de los módulos.

A continuación, con idea de justificar su elección, se enuncian una serie de usos potenciales de la aplicación:

- Es multiplataforma, permite su instalación en Raspbian.
- Muestra los módulos locales y remotos al alcance, independientemente de su configuración.
- Permite configurar los módulos tanto por puerto USB como a distancia.
- Visualiza la topología de la red creada de forma gráfica y tabulada, mostrando cada uno de los nodos y conexiones que la conforman.
- Tiene un abanico de herramientas que permiten desde la recuperación de módulos y pruebas de alcance, hasta cargar actualizaciones de firmware en los módulos de radiofrecuencia.

3.2.4. VNC Viewer

Cómo se explicará en secciones posteriores, la configuración y el uso de las Raspberry Pi precisan de una serie de periféricos qué hacen que su puesta a

punto y su uso posterior sea un poco farragoso. Para simplificar el uso de las SBC, una vez instalado ya su sistema operativo se habilitará el servidor VNC permitiendo conectarse de forma remota a ambas Raspberry Pi desde una aplicación informática llamada VNC Viewer (VNC Viewer, 2020) instalada en una computadora.

3.2.5. ThingSpeak

En un principio, dentro del alcance del proyecto no estaba estipulado el uso de una plataforma de visualización de datos, ya que en un principio el tratamiento y visualización de estos datos se iba a realizar de una forma local, conectando a la Raspberry Pi maestra un monitor por su puerto HDMI.

Sin embargo, durante la consulta de diferentes artículos científicos y de investigación, en donde el alumno realizaba un estudio de los diferentes protocolos de comunicación inalámbrica, fue consciente de la importancia de los sistemas IoT, y quiso añadir dentro de los objetivos del proyecto el uso de un repositorio en la nube como es el caso de ThingSpeak (ThingSpeak, 2020), con la idea en mente de darle más profundidad y calidad al proyecto, y ser a su vez ser un desafío para el alumno.

Según ThingSpeak (2020), el servicio que ofrecen es el de una plataforma analítica para IoT, en la cual permiten subir, visualizar y analizar los flujos de datos en directo. A su vez, provee de una API de MATLAB para realizar análisis y cálculos de los datos a la par que se suben.

En la Figura 18 se resume con una infografía lo expuesto anteriormente.

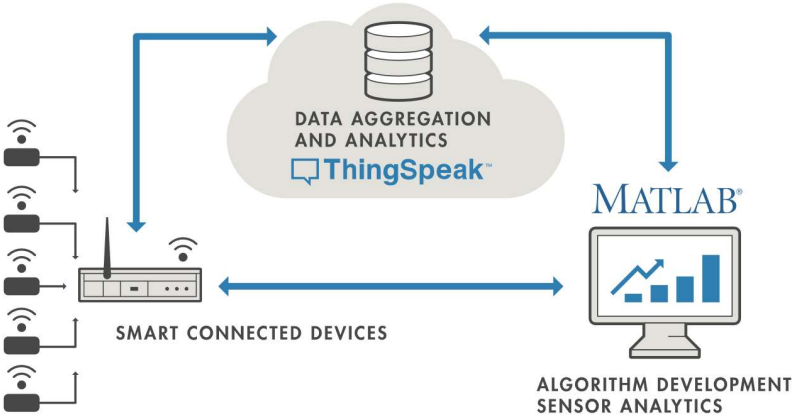


Figura 18: Sistema IoT con ThingSpeak y MATLAB
Fuente: (ThingSpeak, 2020)

Por último, hay que destacar una funcionalidad bastante atractiva que posee esta plataforma, que es la posibilidad de programar reacciones como alarmas y avisos por correo electrónico o Twitter (Twitter, 2020) en función de los datos recabados por los sensores.

4. Desarrollo del proyecto

En este apartado se realiza un análisis detallado de cada uno de los pasos que conforman el proyecto, las dificultades encontradas y las soluciones que se han llevado a cabo.

A continuación, en la Figura 19 se muestra un esquema de todas las partes implicadas en el proyecto con el objetivo de tener un mapa mental esclarecedor de las conexiones y relaciones entre componentes.

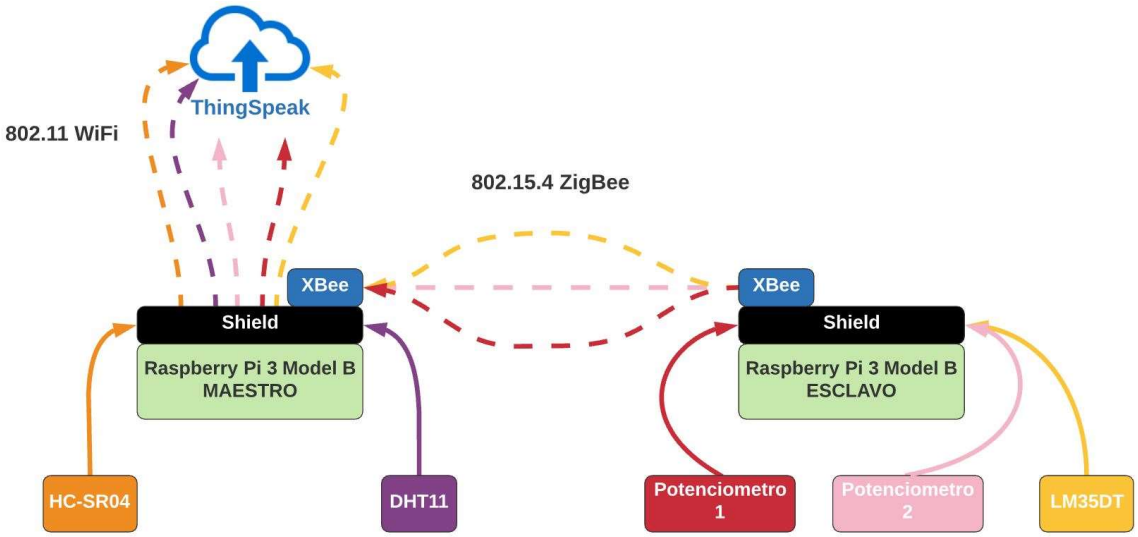


Figura 19: Esquemas de conexiones del proyecto
Fuente: Elaboración propia

En la Figura 19 se aprecia como cada una de las Raspberry Pi (maestra y esclava) recogen los datos de los sensores que tienen conectados. En el caso del esclavo esos datos recogidos son envíados desde su módulo XBee al del maestro mediante el protocolo ZigBee. Finalmente, el maestro sube a ThingSpeak a través de una conexión WiFi, tanto sus datos recogidos por los sensores como los transmitidos por el esclavo, permitiendo su visualización

accediendo a ThingSpeak desde cualquier ordenador o dispositivo con navegador de Internet.

4.1. Montaje y puesta a punto de ambas Raspberry Pi 3 Model B

En esta sección se describen los pasos a seguir para configurar cada una de las Raspberry Pi y su acceso remoto.

4.1.1. Instalación Raspbian

En primer lugar, se formatearon la memoria SD de 32 GB para grabar en ella el sistema operativo. Para ello se utilizó un adaptador de tarjeta y se conectó al ordenador para su formateo.

En la web de la fundación de Raspberry Pi en el apartado de descargas, se elige la opción NOOBS (Raspberry Pi, 2020) que es un gestor de sistemas operativos que permite la instalación de diversas distribuciones Linux (Linux, 2020) en la Raspberry Pi; en esa sección hay una versión Lite y otra completa, ambas son gratuitas y válidas para su descarga.

A continuación, se descomprime NOOBS en la tarjeta SD y se inserta en las Raspberry Pi. Dependiendo de la versión escogida precisará de más conexión de internet para finalizar la instalación.

Luego, se le conecta una pantalla mediante el puerto HDMI, ratón, teclado, un cable de red Ethernet[11] y el cable de alimentación, iniciándose la Raspberry Pi.

Al arrancarse por primera vez crea las particiones necesarias, seguidamente se elige el sistema operativo a instalar, en el caso de este proyecto fue Raspbian; la razón de esta elección se debe a qué trae de base un IDE tanto para Python como C++.

[11] No es necesario el uso de este cable si se tiene acceso a una red WiFi disponible.

4.1.2. Habilitación de acceso remoto y otras interfaces

La Raspberry Pi ofrece dos opciones para su acceso remoto: SSH y VNC. La diferencia ente ambos tipos de interfaces radica, en que en la primera se accede a la Raspberry Pi a través de su terminal, permitiendo tan sólo la ejecución de comandos y programas mediante líneas de comandos. Por otro lado, con el servicio VNC se realiza una conexión en donde se visualiza el escritorio de la Raspberry Pi desde una ventana facilitada por el programa VNC Viewer. Se escogió el uso de la VNC por ser un sistema mucho más cómodo visualmente y a su vez sencillo de usar.

En la Figura 20, se muestra la ventana de configuración[12] de Raspberry Pi, en ella se pueden activar todas las interfaces que sean necesarias. Para el proyecto se habilitaron las que figuran activadas.

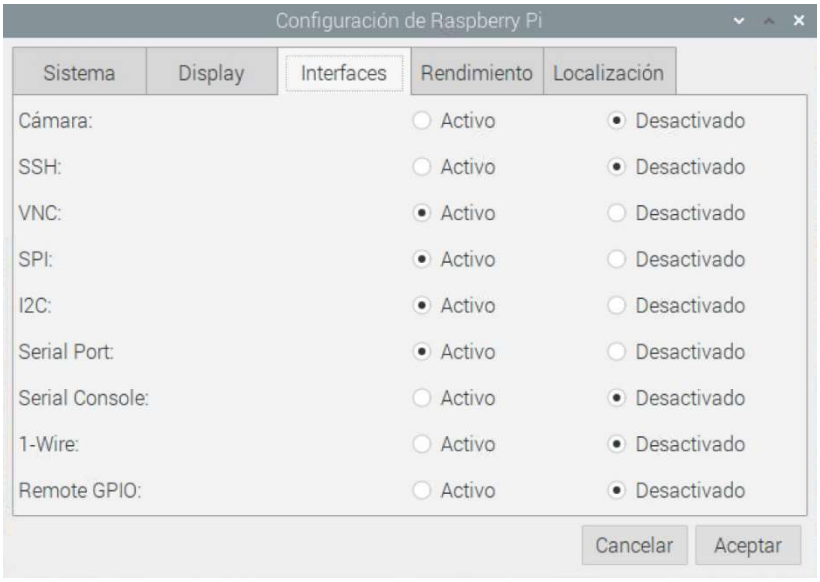


Figura 20: Configuración interfaces Raspberry Pi

4.2. Configuración XBee

A continuación, se detallan los pasos que se realizaron para la configuración de los módulos de radio frecuencia XBee.

[12] Siguiendo esta ruta se accede a ella: Inicio/Preferencias/Configuración de Raspberry Pi/Interfaces.

4.2.1. Uso de XCTU y protocolo ZigBee Peer-to-peer

En primer lugar, se apunta el número serie (Figura 21 parte izquierda) que está en el dorso de cada XBee, se usará más tarde como parámetro (SH y SL), seguidamente, se conectan las XBee al puerto USB mediante un adaptador llamado XBee Explorer (Sparkfun, 2020), como se observa en la parte derecha de la Figura 21.

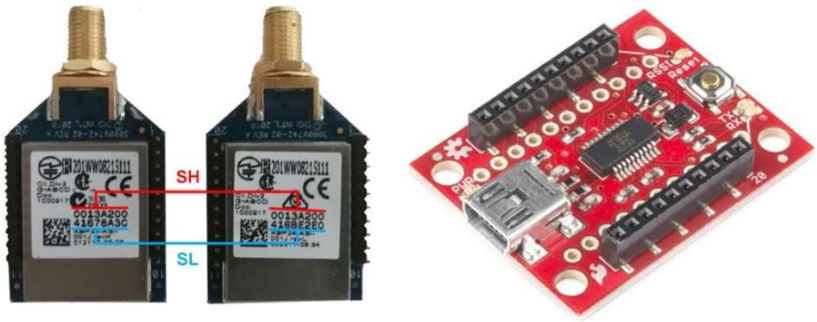


Figura 21: XBee Explorer

En segundo lugar, se arranca el software XCTU y se selecciona la opción de añadir un módulo . Seleccionamos el módulo encontrado (Figura 22), se actualiza si fuera necesario el firmware mediante el botón . Como se observa en la Figura 22, ambas XBee tienen disponible una versión nueva de firmware, pero por motivos de estabilidad se decidió la versión anterior.

Figura 22: Módulos XBee, maestro y esclavo respectivamente

4.2.2. Protocolo ZigBee peer-to-peer

A continuación, se van a mostrar los parámetros a configurar en cada una de la XBee para generar una red punto a punto. No obstante, se va a definir de qué forma a partir de una red punto a punto se puede expandir hasta formar una red mallada mediante la adición de más módulos XBee.

En la Tabla 8 se exponen los parámetros a introducir para la red punto a punto.

Tabla 8: Configuración de red punto a punto
Fuente: Elaboración propia

Parámetro	Maestro	Esclavo	Definición
CH	C	C	Canal
ID	D161	D161	PAN ID
DH	0013A200	0013A200	Dirección de destino alta
DL	416BE2E0	41678A3C	Dirección de destino baja
PL	1	1	Nivel de potencia
CE	1	0	Coordinador activado

Nótese que si se quisiera añadir dispositivos adicionales trabajando como esclavos usaría los mismos parámetros que el esclavo actual. En el caso de una configuración como router dentro de la red, tendría que cargarse el firmware adecuado. Para finalizar, la Figura 23 esclarece el concepto de los parámetros.

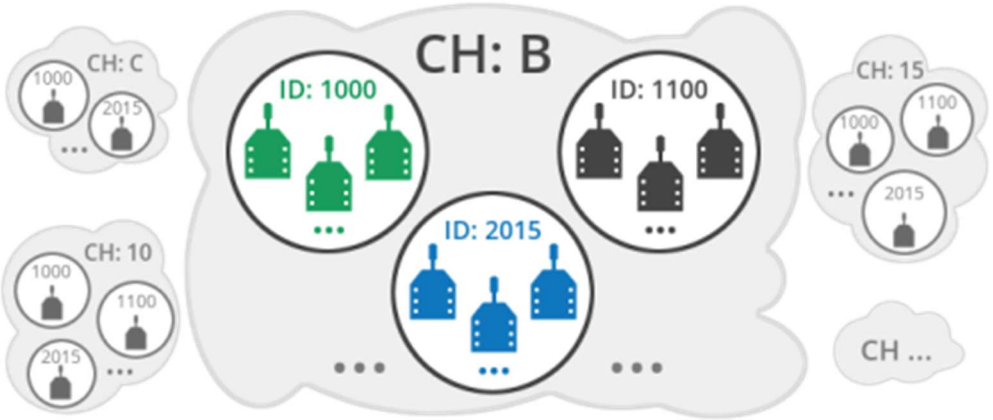


Figura 23: Relación de redes según parámetros
Fuente: (Aprendiendo Arduino, 2017)

4.3. Conexionados de sensores

En este subcapítulo se aborda todo el apartado técnico referente a la conexión de los sensores que conforman el proyecto. Se expondrá en una primera instancia las consideraciones a tener en cuenta cuando se trabaja con determinado tipo de sensores y tensiones. Por otro lado, se detallarán los esquemas de conexionado.

4.3.1. Protección de la placa

Es importante destacar que las entradas digitales de la Raspberry Pi admiten una tensión de 3,3V y no son tolerantes a una tensión mayor. Ni la Raspberry Pi ni el Shield de Coocking Hacks disponen de un sistema de protección contra sobretensiones. Es decir, las entradas digitales usan un nivel lógico de 3,3 voltios y no toleran una tensión mayor, como por ejemplo es el caso de Arduino que sí admite tensiones a 5V.

No obstante, se puede trabajar con sensores qué precisen de 5 voltios para su alimentación, siempre y cuando el envío de la medida a la Raspberry Pi tenga una tensión máxima de 3,3V. Para ello, una solución sencilla es realizar un divisor de tensión, como se ejemplifica en la Figura 24.

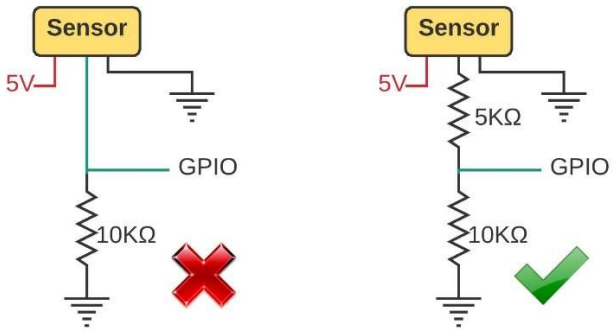


Figura 24: Circuito sin y con divisor de tensión

El cálculo siguiente justifica lo expuesto en la Figura 24:

$$v_{GPIO} = 5V \cdot \frac{10k\Omega}{10k\Omega + 5k\Omega} = 3{,}3V$$

4.3.2. Planos de conexionado

En este epígrafe se detallan los esquemas de conexiones de cada una de las Raspberry Pi 3 realizados con el software Fritzing (Fritzing, 2020).

En la Figura 25 se muestra el esquema de conexiones del maestro que actúa con el rol de coordinador de la red.

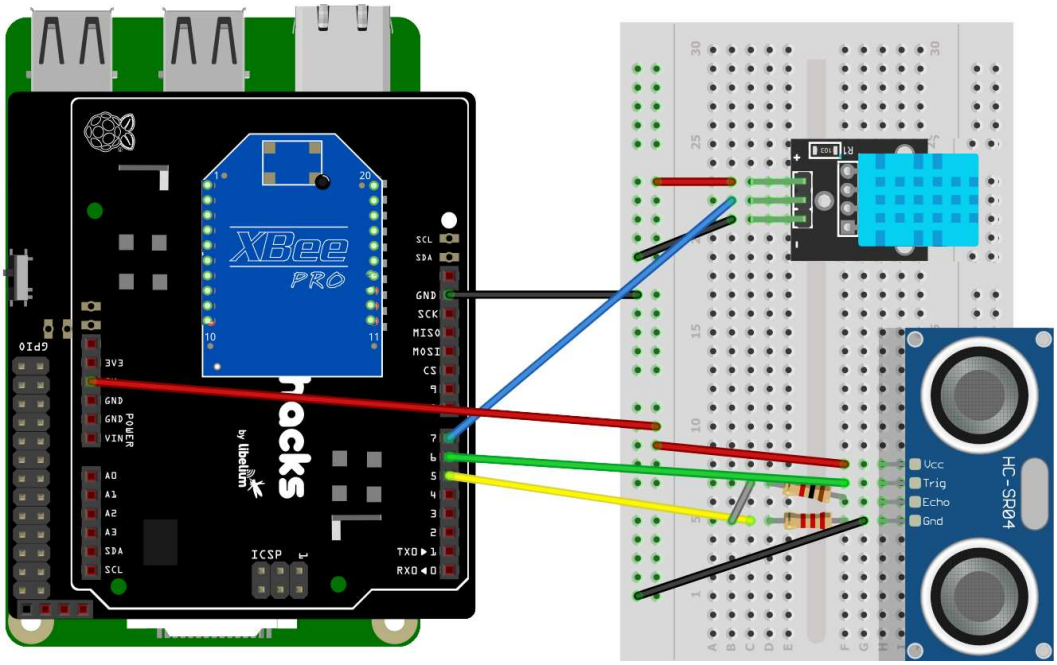


Figura 25: Conexiones en nodo maestro

Para complementar la Figura 25, se ha generado la Tabla 9, informando de las conexiones sensor-Raspberry Pi.

Tabla 9: Conexiones en nodo maestro

Fuente: Elaboración propia

Pin del sensor	Pin en shield Coocking Hacks
DHT11	
Vcc	5V
Data	7
GND	GND
HC-SR04	
Vcc	5V
Trigger	6
Echo	5
GND	GND

Siguiendo la misma estructura anterior, en la Figura 26 se detalla el esquema de conexiones del esclavo que actúa con el rol de dispositivo final de la red.

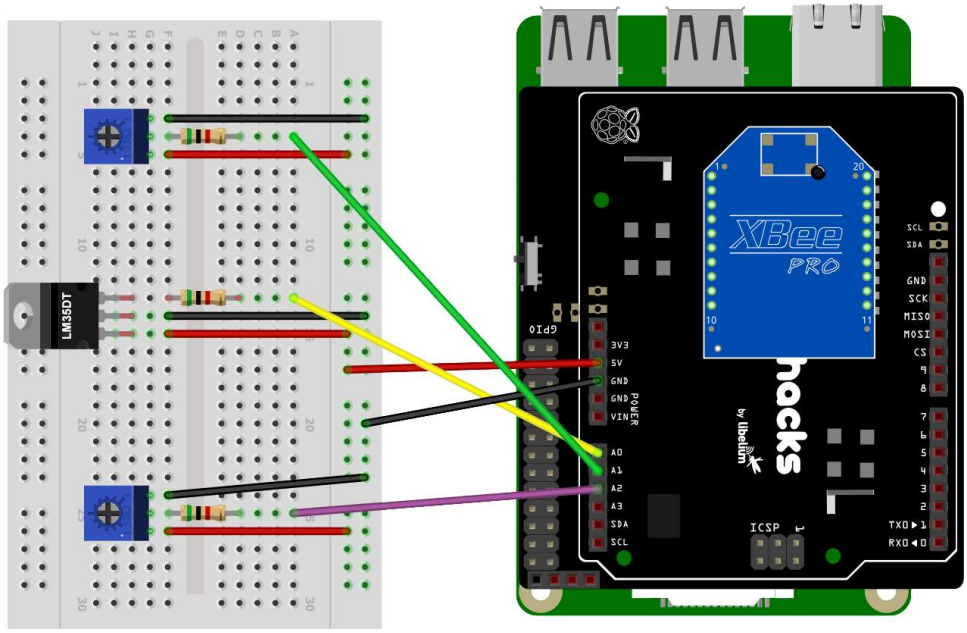


Figura 26: Conexiones en nodo esclavo

Finalmente, en la Tabla 10 se exponen las conexiones sensor-Raspberry Pi.

Tabla 10: Conexiones en nodo esclavo
Fuente: Elaboración propia

Pin del sensor	Pin en shield Coocking Hacks
LM35DT	
Vcc	5V
Data	A0
GND	GND
Potenciómetros	
Vcc	5V
Data	A1 / A2
GND	GND

4.4. Programación

La progrmacion de la rPi esta organizada de a la siguiente forma. En el eclavao hay un afuncion que lee lo sensore y en l maestra lo que hace. Luego se mostrará los pseudocodigos, pero antes de eso se hará una explicaicon de ardupi que es necesaria para el esclavoEn este subcapítulo se describe el uso de la librería arduPi, además de los pseudocódigos de los programas realizados en cada una de las Raspberry Pi para la lectura de los sensores y el envío de datos a ThingSpeak.

4.4.1. Librería arduPi

Como se introdujo en capítulos anteriores, arduPi es una librería de C++ que facilita la escritura de programas en la Raspberry Pi, utilizando la misma sintasis de Arduino. Su potencial y la causa de por qué se usó, radica en que permite utilizar casi todas[13] las funciones que controlan el puerto serie, SPI, I2C y los GPIO con la sintasis de Arduino.

Desde la web de Coocking Hacks se puede descargar esta librería, la cual viene acompañada por los siguientes archivos: *arduPi.h*, *arduPi.cpp* y *arduPi.cpp*. Este último archivo es una plantilla que sirve de inicio para escribir programas e incluso permite volcar cualquier programa realizado en Arduino.

Para concluir, al ser arduPi una librería de C++ se usará g++[14] para compilarla, y una vez compilada, se obtiene un archivo objeto (.o). Este archivo se adjuntará a la hora de compilar el programa deseado incluyéndose en él la librería arduPi de la siguiente manera:

```
>>g++ MI_PROGRAMA_FUENTE.cpp arduPi.o -o MI_PROGRAMA_EJECUTABLE
```

4.4.2. Pseudocódigos

EnvioAMaestro: El programa de la Raspberry Pi esclava para la lectura de los sensores (EnvioAMaestro.cpp) se detalla mediante un pseudocódigo para una mejor comprensión:

1. Cargar librerías
2. Inicializar variables y pines de entrada y salida
3. Colocar pin 3 en High para habilitar socket de la XBee[15]
4. Inicializar comunicación serial y establecer el baud rate

[13] Es importante remarcar el 'casi todas', ya que como se verá en la sección de solución de errores la librería arduPi no implementa todas las funciones de comunicación con el puerto serie.
[14] Es una orden que sirve para compilar y enlazar programas en C++.
[15] En la sección de troubleshooting se explica a fondo el por qué de esta acción.

5. While(1)

   %%Lectura del sensor de temperatura

   a. Dato=Read(sensor LM35DT) y cálculo temperatura
   b. Send_serial(dato) y Send_serial(\n) por puerto serie[16]
   c. Printf(dato)
   d. Sleep(20 segundos)

   %%Lectura del potenciómetro 1, pasos análogos al de lectura del sensor de temperatura, excepto Dato=Read(potenciómetro 1)

   %%Lectura del potenciómetro 2, pasos análogos al de lectura del sensor de temperatura, excepto Dato=Read(potenciómetro 2)

La Raspberry Pi irá leyendo datos de cada uno de sus sensores empezando por el sensor de temperatura LM35DT hasta el segundo potenciómetro, con intervalo de 20 segundos entre cada sensor. En cuanto al código EnvioAMaestro.cpp se encuentra en el Anexo V.

TSLecturaEsclavo: Por otro lado, se muestra el pseudocódigo de lectura de puerto serie y subida de estos datos a ThingSpeak por parte del maestro, llamado TSLecturaEsclavo.py

1. Cargar librerías
2. Inicializar puerto serie
3. Definir código de la API de ThingSpeak
4. While(Serial_dato(1))
   a. Serial_Read(Serial_dato)
   b. Descodificar eliminando carácter inicial y espacios
   c. Dato=Serial_dato
      i. If (0>dato<100) %%Caso sensor temperatura
         1. Temp=Dato
         2. Iniciar petición servidor ThingSpeak y subir dato
         3. Print(Temp, respuesta_servidor)

[16] Recuérdese que Xbee se comunica por el puerto serie

4. Close_ThingSpeak
5. Esperar 0,5 segundos

%%Caso potenciómetro 1

i. If (101>dato<281)

%%pasos análogos al de lectura del sensor de temperatura, excepto Pot1=Dato

%%Caso potenciómetro 2

i. If (282>dato<462)

%%pasos análogos al de lectura del sensor de temperatura, excepto Pot2=Dato

d. Sleep(7,5 seg)

Según el valor leído en el puerto serie, la Raspberry Pi maestra filtra el dato y lo envía al campo asignado en ThingSpeak. Sobre TSLecturaEsclavo.py se encuentra desarrollado en el Anexo VI.

TSDHT11: El pseudocódigo que describe el programa (TSDHT11.py) del maestro para la lectura y subida del sensor de humedad y temperatura, DHT11, es el siguiente:

1. Cargar librerías
2. Inicializar variables y pines de entrada y salida
3. Definir código de la API de ThingSpeak
4. While(1)
   a. Read(humedad, temperatura)
   b. Iniciar petición servidor ThingSpeak y subir dato
   c. Print(Temp, respuesta_servidor)
   d. Close_ThingSpeak
   e. Sleep(12 seg)

Disponible el código fuente en el Anexo VII

TSultrasonido: el programa (TSultrasonido.py) de la Raspberry Pi maestra que lee el sensor de ultrasonido HC-SR04 se define por el siguiente pseudocódigo:

1. Cargar librerías

2. Inicializar pines de entrada y salida
3. Definir código de la API de ThingSpeak
4. While(1)
    a. Sleep(2 seg)
    b. Trigger(ON) y Set_time()
    c. ECHO(ON) y Stop_time()
    d. Time= Set_time()-Stop_time()
    e. Distancia=calculo
    f. Iniciar petición servidor ThingSpeak y subir dato
    g. Subir dato
    h. Print(Temp, respuesta_servidor)
    i. Close_ThingSpeak
    j. Sleep(12 seg)

Disponible el código fuente en el Anexo VIII

4.5. ThingSpeak

El objeto de este subcapítulo es el de explicar cómo funciona la plataforma y su potencial como herramienta para proyectos de IoT.

4.5.1. Creación de cuenta

Para generar una cuenta en ThingSpeak es necesario tener una cuenta en MathWorks. Para ello, ThingSpeak te permite crear una cuenta de MathWorks que una vez realizada se vinculará a la plataforma para su acceso. En la Figura 27 se muestra el perfil de la cuenta usada en el proyecto, en la cual es especialmente important señalar el campo API key.

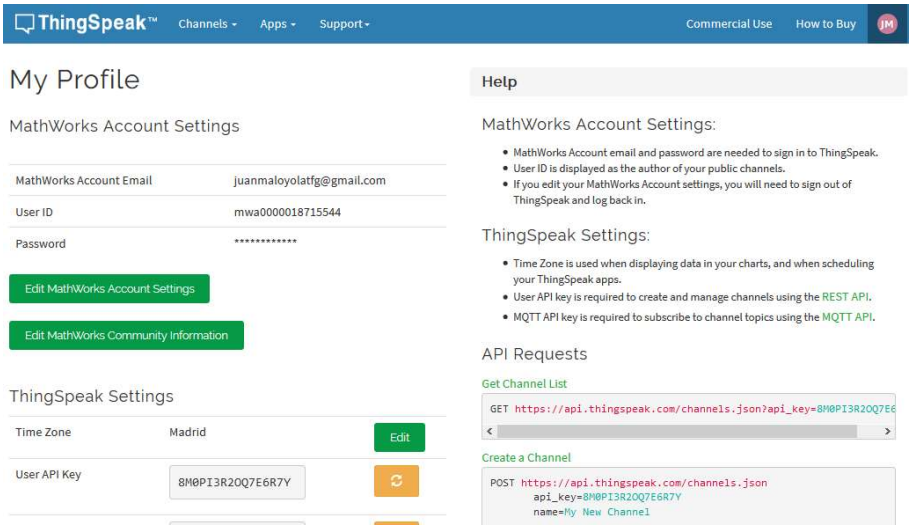


Figura 27: Cuenta de usuario del proyecto

4.5.2. Generación de canal

La plataforma te permite tener hasta 4 canales por cuenta, cada uno de ellos dispone de 8 campos para subir datos. Tan solo hay que seleccionar la opción de 'Mis canales', luego a 'Nuevo Canal' y rellenar los campos deseados de la Figura 28. Es importante mencionar que el campo 'Field' junto con 'API KEY' (disponible en el perfil de la cuenta) son los parámetros principales para el envío de datos a la plataforma.

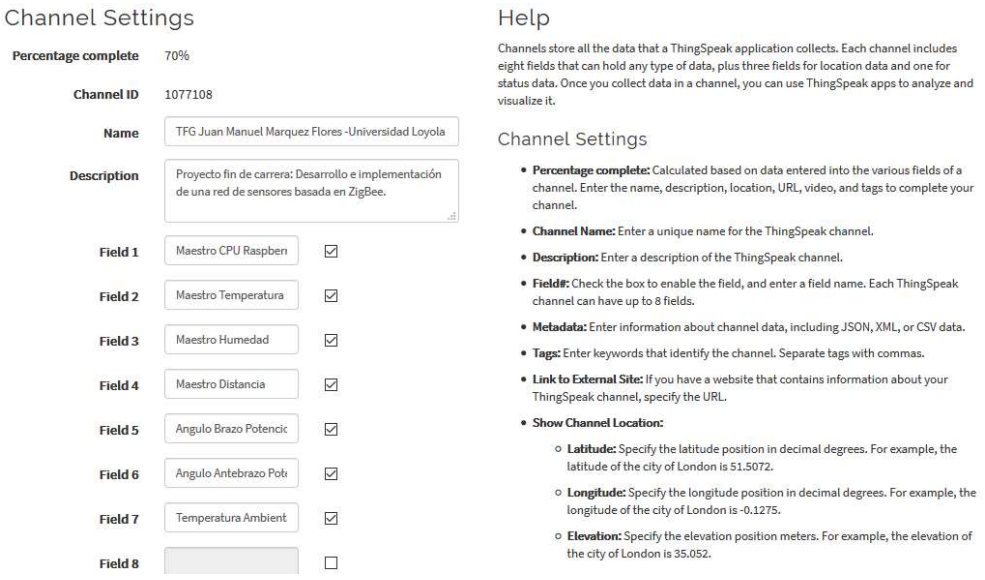


Figura 28: Configuración de canal en ThingSpeak

4.5.3. Importación, visualización de datos y exportación

Con el canal ya generado, tan solo se necesita añadir a cada programa las líneas de código necesarias para la subida de datos. La plataforma te permite tres métodos para ello: HTTP, MQTT y desde MATLAB a través de una función generada para tal efecto. En cuanto al proyecto, se eligió HTTP frente a MQTT, debido al tipo de variables que se están midiendo, no se precisa de mucha velocidad de refresco y por esta razón se eligió HTTP

En cuanto a la visualización de los datos, tan solo hay que acceder a cualquiera de las dos pestañas para ello: canal privado o público. La diferencia entre ambos radica en que al primero tan solo tiene acceso el propietario y éste decide qué es visible de este canal en el público. En la Figura 29 se muestra el canal privado del proyecto.

Figura 29: Visualización de datos del proyecto en ThingSpeak

Finalmente, en la pestaña de importación o exportación, se da la opción de subir o descargar los datos en un archivo CSV.

4.5.4. Alarmas, notificaciones y APIs

Una de las capacidades más atractivas y útiles que tiene ThingSpeak son las *Apps*. En el caso del proyecto, se usó la aplicación *React*, a través de ellas es posible mandar reacciones (en función de los datos ingresados, es decir, si cualquier sensor sobrepasa una determida cota, mandará un mensaje a la plataforma que se escoja a través de sus *Apps* (ThingHTTP, ThingTweet, y MATLAB Analysis)

Con relación a la idea anterior, en el proyecto se han realizado 4 alarmas que al activarse publican un *tweet* si se cumplen alguna de las siguientes condiciones:

- DHT11: Humedad relativa por encima de 45%.
- DHT11: Humedad relativa por debajo de 25%.
- DHT11: Temperatura ambiente por encima de 40 ºC.
- LM35DT: Temperatura ambiente por encima de 35ºC.

En la Figura 30, se muestra un ejemplo de una reacción de humedad relativa por encima de lo permitido en el lado del nodo maestro.

Figura 30: Reacción enviada a Twitter por alta humedad relativa

5. Conclusiones

5.1. Consecución de objetivos

Partiendo del objetivo global, que consistía en el despliegue de una red de sensores y su comunicación mediante el protocolo ZigBee, se puede afirmar con total seguridad que se ha cumplido el fin principal, e incluso se ha incorporado algunas funcionalidades adicionales.

Dados los objetivos citados en el apartado '1.2 Objetivos', se concluye:

1. Que el objetivo principal, montaje de un sistema IoT donde una Rapsberry Pi actúa como coordinador central y otra como nodo, se ha realizado con éxito.
2. Que se ha conseguido utilizar sensores analógicos en la Raspberry Pi gracias a la implementación de un *Shield*.
3. Que se ha elegido el protocolo y la topología de red idónea para el uso de las Xbee.
4. Que los datos puedan ser visualizados en *stream*, gracias a ThingSpeak, sin depender de un sistema físico, como por ejemplo el uso de un monitor.
5. Que el presente documento ayude a profesionales de la ingeniería a desarrollar un sistema IoT o ser utilizado de consulta para futuros proyectos.

Adicionalmente, se han conseguido otros logros que le da más profundidad al proyecto:

1. Que se ha podido usar sensores digitales adicionalmente a los analógicos que se citaron anteriormente.

2. Que se ha programado alertas que informan posibles problemas en el sistema.

Finalmente, hay que destacar que el proyecto no se ha desarrollado sólo dentro de un marco teórico, si no que además ha sido implementado en la realidad y es totalmente funcional.

5.2. Posibles mejoras y objetivos futuros

Dentro de las posibles mejores que puede tener el sistema, se destaca:

- Que se puede incluir diferentes sensores al sistema, como por ejemplo cámaras, con idea de generar un sistema de videovigilancia.
- Que se investigue el uso e implementación de actuadores que den otras aplicaciones dentro del sistema complementando el uso de sensores.
- Que con objetivo de ampliar la cobertura de la red se incorporen, adicionalmente, más nodos al sistema, añadiendo otras Raspberry Pi. De esta manera será posible recoger información cubriendo un área mayor.
- Que se pueda cambiar la topología de malla que permita realizar una red mallada o de tipo árbol. Esto provocaría una intercomunicación más avanzada que daría lugar a otras funcionalidades no cubiertas actualmente.
- Que se depuren los códigos de los programas que gobiernan cada nodo, provocando una mejora del sistema y del funcionamiento.

Respecto a los posibles objetivos que se puedan desarrollar en un futuro:

- Que partiendo de la base de este proyecto se realice un sistema de monitorización de los laboratorios de la Escuela Técnica Superior de Ingeniería de la Universidad Loyola Andalucía, que sirva como herramienta de apoyo para el seguimiento de las clases prácticas de manera remota.
- Que se desarrolle una herramienta para reconocer los espacios disponibles en un aparcamiento, y ser informado de esta disponibilidad a través de una *App* instalada en el *smartphone*.

- Que se genere un sistema automático de riego, iluminación y temperatura en función de las necesidades de cada una de las plantas que conformen el invernadero. Se incluiría en este sistema alarmas que avisarían del estado de cada planta.

5.3. Conclusiones personales

En última instancia, se concluye que el internet de las cosas no solo se puede implementar en la Industria, sino que también en ámbitos cotidianos, dando respuestas a necesidades actuales o futuras, o simplemente facilitando la vida de las personas. Se afirma que es un futuro en progreso, ya que vivimos en un mundo cada vez más interconectado, por lo que existirán múltiples funcionalidades que harán la vida mucho más sencilla. Por ello, es el momento de invertir o desarrollar sistemas IoT, sobre todo, en ámbitos que no están cubiertos por esta tecnología o que se han quedado desfasados en el tiempo. El alumno ha podido desarrollar un sistema funcionar IoT de manera autónoma gracias a los conocimientos adquiridos en asignaturas tales como, Electrónica o Informática. El desafío de desarrollar el proyecto en cuarentena ha dado lugar a experimentar lo útil que habría sido para los alumnos un sistema en el laboratorio, como el explicado a lo largo de esta memoria, para la realización de las prácticas, por lo que no hay mejor camino para desarrollar una idea que el sentir que es necesaria.

6. Anexos

Anexo I: Troubleshooting

En esta sección se muestra cómo se resolvieron aquellos problemas que fueron cruciales para la correcta finalización del proyecto.

Digital Switch placa Coocking Hacks v2.0

La característica principal que tiene el shield de Coocking Hacks es que trae un interruptor digital para encender o apagar el módulo inalámbrico, es decir, no basta con tan solo ensamblar el shield a la Raspberry Pi y usarlo, si no que precisaba de activar este interruptor digital para poder hacer uso de, por ejemplo, el XBee en el caso del proyecto.

Para resolver este problema existen dos vías, o bien se coloca un cable alimentando el pin digital 3, o bien, se escribe un par de líneas de código en donde se coloca en High ese pin.

Limitaciones de la Librería arduPi

La librería arduPi presentaba limitaciones en el uso de la comunicación serie y algunas funciones estaban mal implementadas, dificultando así la escritura de programas:

- La función *Serial.println()* original de Arduino incluye un retorno de carro al final de la cadena de caracteres, sin embargo, la función de arduPi no lo incluía.

  El problema fue resuelto anidando el comando *Serial.print()* para enviar el salto de carro.

- Otro inconveniente que había al usar la librería es el hecho de que no incluía la función *Serial.readstring()*, impidiendo la lectura de más de un carácter en el puerto serie.

  Se resolvió utilizando Python como lenguaje de lectura en el extremo del nodo maestro.

Habilitación UART en Raspberry Pi 3 Model B

La Raspberry Pi 3 Model B tiene dos puertos UART: uno dedicado al Bluetooth y otro para los pines GPIO. Para configurar los pines de comunicación serie hay que realizar los siguientes pasos:

1. Lo primero que se ha de hacer es habilitar la comunicación serie desde el menú de configuración de Raspbian e inhabilitar el *login Shell* para evitar que la Raspberry Pi al iniciarse envíe datos por la UART.
2. Tras esto, hay que configurar el archivo de inicio del sistema, y añadirle las siguientes líneas de código:

```
>>core_freq=250

>>enable_uart=1

>>dtoverlay=pi3-miniuart-bt
```

De esta forma, se habrá solucionado por orden los siguientes problemas: ajustar el reloj de la UART, activar la UART y permitir que el Shield de Coocking Hacks tenga acceso a la UART.

Anexo II: Presupuesto del proyecto

A continuación, se detalla el coste total de la implementación de este trabajo de fin de grado.

Realizado por: Juan Manuel Marquez Flores

Proyecto: Desarrollo e implementación de una red de sensores para monitorización basada en ZigBee.

Duración: 5 meses – 300 horas en total

Software

Descripción	Ud.	Cantidad	Precio	Importe
Thonny	NA	1	- €	- €
ThingSpeak	NA	1	- €	- €
Raspbian	NA	1	- €	- €
Geany	NA	1	- €	- €
XCTU	NA	1	- €	- €
VNC Viewer	NA	1	- €	- €
Fritzing	NA	1	- €	- €
			Subtotal	**- €**

Hardware

Descripción	Ud.	Cantidad	Precio	Importe
Raspberry Pi 3 Model B	Euros	2	34,00 €	68,00 €
XBee Explorer USB	Euros	1	27,95 €	27,95 €
Monitor Acer 19.5" LED	Euros	1	64,00 €	64,00 €
Teclado GENIUS KB-116 USB	Euros	1	9,99 €	9,99 €
Ratón Logitech B100	Euros	1	4,99 €	4,99 €
Tarjeta micro SD 32 GB	Euros	2	5,15 €	10,30 €
Raspberry Pi to Arduino Shield	Euros	2	27,40 €	54,80 €
XBee Pro S1	Euros	2	38,61 €	77,22 €
Antena RP-SMA 2.4 GHz	Euros	2	8,95 €	17,90 €
Sensor LMT35DT	Euros	1	3,68 €	3,68 €
Sensor HC-SR04	Euros	1	1,95 €	1,95 €
Sensor DHT11	Euros	1	6,99 €	6,99 €
Potenciómetro 3326P-1-103	Euros	2	1,11 €	2,22 €
Protoboard	Euros	2	4,95 €	9,90 €
Cargador micro USB	Euros	2	9,99 €	19,98 €
Kit de jumpers y resistencias	Euros	1	5,30 €	5,30 €
			Subtotal	**385,17 €**

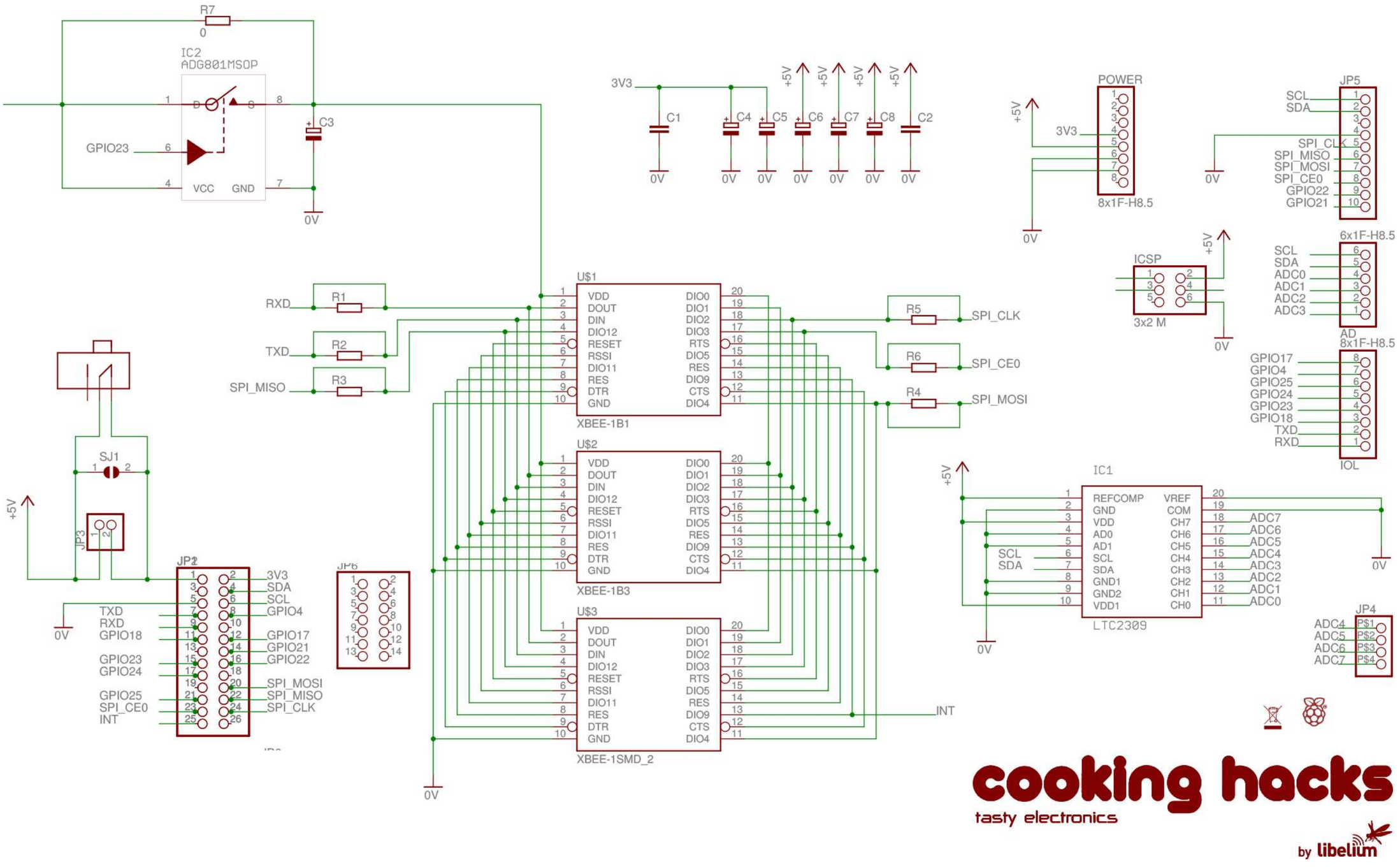
Mano de obra				
Descripción	**Ud.**	**Cantidad**	**Precio**	**Importe**
Diseño programa nodo esclavo	horas	30	40,00 €	1.200,00 €
Diseño programas nodo maestro	horas	45	40,00 €	1.800,00 €
Diseño comunicación protocolo ZigBee	horas	40	40,00 €	1.600,00 €
Programación ThingSpeak	horas	25	40,00 €	1.000,00 €
Diseño, elección y montaje de componentes	horas	50	40,00 €	2.000,00 €
Ingeniería y puesta a punto	horas	110	40,00 €	4.400,00 €
			Subtotal	**12.000,00 €**

Presupuesto total	
Software	0,00 €
Hardware	385,17 €
Mano de obra	12.000,00 €
	12.385,17 €
Gastos generales (13%)	1.610,07 €
Beneficio industrial (6%)	743,11 €
	14.738,35 €
IVA (21%)	3.095,05 €
Total presupuesto	**17.833,41 €**

El presupuesto total asciende a la cantidad de diecisiete mil ochocientos treinta y tres con cuarenta y un euros (17.833,41€)

En Sevilla, a 13 de julio de 2020 Firmado: Juan Manuel Márquez Flores

Anexo III: Esquema eléctrico Raspberry Pi to Arduino Shield



Anexo IV: Código plantilla arduPi, arduPi_template.cpp

```
//Incluye librería arduPi
  #include "arduPi.h"

  /*****************************************************************
   *  SI TU CODIGO DE ARDUINO TIENE OTRAS FUNCIONES APARTE DE  *
   *  setup() Y loop() DEBES DECLARARLAS AQUÍ                  *
   * ***************************************************************/

  /*****************************
   * TU CODIGO DE ARDUINO AQUI *
   * ***************************/

  int main (){
    setup();
    while(1){
      loop();
    }
    return (0);
  }
```

Anexo V: Código esclavo, EnvioAMaestro.cpp

```
/*************************************************
*         PROYECTO FINAL DE CARRERA 2020         *
*************************************************
* Titulo: Desarrollo e implementacion de una red de *
* de sensores para monitorizacion basada en ZigBee  *
*************************************************
* Nombre:  EnvioAMaestro.cpp                     *
* Autor:   Juan Manuel Marquez Flores            *
* Carrera: Grado en Ingenieria Electromecanica   *
* Entidad: Universidad Loyola Andalucia          *
*************************************************/
//Incluye libreria arduPi
#include "arduPi.h"
//Declaracion de variables
int tempPin0 = 0;
float temperatura = 0;
int potPin1 = 1;
float BAngulo = 0;
int potPin2 = 2;
float ABAngulo = 0;
//Desbloquear shield Coocking Hacks e inicia puerto serie
void setup(){
  pinMode(3,OUTPUT);
  digitalWrite(3,HIGH);
  Serial.begin(9600);
}
//Comienzo del bucle, parada por teclado
void loop(){
  //Lectura y envio sensor LM35DT
  float val0 = 0;
  val0 = analogRead(tempPin0);
  temperatura = (5*(val0)*100/1024);//De señal a temperatura
  digitalWrite(3,HIGH);//Asegura que el modo inalambrico sigue activo
  Serial.print(temperatura,0);
  Serial.print("\n");
  Serial.flush();//Espera que la transmision de datos finalize
  printf("Temperatura: %.1f \n",temperatura);
  delay(20000);
  //Lectura y envio potenciometro
  int val1 = 0;
  val1 = analogRead(potPin1);
  BAngulo = ((val1*180)/1020)+101;//Señal a angulo y codificacion
  digitalWrite(3,HIGH);
  Serial.print(BAngulo,0);
  Serial.print("\n");
```

```
  Serial.flush();
  printf("Angulo brazo: %.0f \n",BAngulo-101);
  delay(20000);
  //Lectura y envio potenciometro 2
  int val2 = 0;
  val2 = analogRead(potPin2);
  ABAngulo = ((val2*180)/1020)+282;//Señal a angulo y codificacion
  digitalWrite(3,HIGH);
  Serial.print(ABAngulo,0);
  Serial.print("\n");
  Serial.flush();
  printf("Angulo antebrazo: %.0f \n",ABAngulo-282);
  delay(20000);

}

int main (){
  setup();
  while(1){
    loop();
  }
  return (0);
}
```

Anexo VI: Código maestro lectura esclavo y TS, TSLecturaEsclavo.py

```
#######################################################
#         PROYECTO FINAL DE CARRERA 2020              #
#######################################################
# Titulo: Desarrollo e implementacion de una red de #
# de sensores para monitorizacion basada en ZigBee  #
#######################################################
# Nombre:  TSLecturaEsclavo.py                        #
# Autor:   Juan Manuel Marquez Flores                 #
# Carrera: Grado en Ingenieria Electromecanica        #
# Entidad: Universidad Loyola Andalucia               #
#######################################################
import httplib
import urllib
import serial
import string
import time
# Inicializa puerto serie, conexion con esclavo
esclavo = serial.Serial('/dev/ttyAMA0', 9600, timeout=1)
# Clave unica para subir a canal de ususario de ThingSpeak
key = "6EAM55ROJ4R4L319"
# Se define la funcion TSlector
def TSlector():
    if (esclavo.inWaiting()>0): # Mientras haya datos en puerto serie
        raw = str(esclavo.readline())# Toma el dato del puerto serie
        # Le Quitamos caracteres: ', b del estring y espacios
        filtrado = raw.replace('b','').replace("'",' ').strip()
        # Dato ya descodificado
        valor = float(filtrado)
        # En este caso si el valor esta en el rango, sera temperatura
        if  0.0 <= valor <= 100.0:
            temperatura = float(valor)
            # Comienza el proceso de conexion con ThingSpeak
            params = urllib.urlencode({'field7': temperatura, 'key':key
})
            headers = {"Content-typZZe": "application/x-www-form-
urlencoded","Accept": "text/plain"}
            conn = httplib.HTTPConnection("api.thingspeak.com:80")
            # Intenta publicar el dato haciendo peticion al servidor
            try:
                conn.request("POST", "/update", params, headers)
                response = conn.getresponse()
                print("Temperatura: "  ,temperatura)
                print response.status, response.reason
                data = response.read()
                conn.close()
```

```
            except:
                print "Conexion fallida"
            time.sleep(.5)
        elif 101 <= valor <= 281:
            BAngulo = valor-101
            BAngulo = int(BAngulo)
            params = urllib.urlencode({'field5': BAngulo, 'key':key })
            headers = {"Content-typZZe": "application/x-www-form-
urlencoded","Accept": "text/plain"}
            conn = httplib.HTTPConnection("api.thingspeak.com:80")
            try:
                conn.request("POST", "/update", params, headers)
                response = conn.getresponse()
                print("Angulo brazo: " ,BAngulo)
                print response.status, response.reason
                data = response.read()
                conn.close()
            except:
                print "Conexion fallida"
            time.sleep(.5)

        elif 282 <= valor <= 462:
            ABAngulo = valor-282
            ABAngulo = int(ABAngulo)
            params = urllib.urlencode({'field6': ABAngulo, 'key':key })
            headers = {"Content-typZZe": "application/x-www-form-
urlencoded","Accept": "text/plain"}
            conn = httplib.HTTPConnection("api.thingspeak.com:80")
            try:
                conn.request("POST", "/update", params, headers)
                response = conn.getresponse()
                print("Angulo brazo: " ,ABAngulo)
                print response.status, response.reason
                data = response.read()
                conn.close()
            except:
                print "Conexion fallida"
            time.sleep(.5)
# Salvo que se interrumpa por teclado o haya error, no se interrumpira el
bucle
if __name__ == "__main__":
        while True:
                TSlector()
                time.sleep(19)
```

Anexo VII: Código maestro lectura DHT11 y TS, TSDHT11.py

```
#######################################################
#         PROYECTO FINAL DE CARRERA 2020              #
#######################################################
# Titulo: Desarrollo e implementacion de una red de #
# de sensores para monitorizacion basada en ZigBee  #
#######################################################
# Nombre:  TSDHT11.py                                 #
# Autor:   Juan Manuel Marquez Flores                 #
# Carrera: Grado en Ingenieria Electromecanica        #
# Entidad: Universidad Loyola Andalucia               #
#######################################################
import httplib
import urllib
import time
import RPi.GPIO as GPIO
import dht11
import datetime
# Inicializa GPIO y numeracion segun BCM
GPIO.setwarnings(True)
GPIO.setmode(GPIO.BCM)
# Declaracion de pines GPIO
instance = dht11.DHT11(pin=17) #Pin 7 en el Shield de Coocking Hacks
# Clave unica para subir a canal de ususario de ThingSpeak
key = "6EAM55ROJ4R4L319"
# Se define la funcion DHT11
def DHT11():
    while True: # Ciclo infinito, para terminarlo pulsar Ctrl+C
        GPIO.setwarnings(True)
        GPIO.setmode(GPIO.BCM)
        result = instance.read()
        if result.is_valid():
            temperatura = result.temperature
            humedad = result.humidity
            # Comienza el proceso de conexion con ThingSpeak
            params = urllib.urlencode({'field2': temperatura, 'field3':
humedad, 'key':key })
            headers = {"Content-typZZe": "application/x-www-form-
urlencoded","Accept": "text/plain"}
            conn = httplib.HTTPConnection("api.thingspeak.com:80")
            # Intenta publicar el dato haciendo peticion al servidor
            try:
                conn.request("POST", "/update", params, headers)
                response = conn.getresponse()
                print("Temperatura: %-3.1f C" % temperatura)
                print("Humedad: %-3.1f %%" % humedad)
```

```
                print (response.status, response.reason)
                data = response.read()
                conn.close()
            except:
                print ("connection failed")
            break
# Salvo que se interrumpa por teclado o haya error, no se interrumpira el
bucle
if __name__ == ("__main__"):
    try:
        while True:
                DHT11()
                time.sleep(15)
    except KeyboardInterrupt:
        print("Cleanup")
        GPIO.cleanup() # Libera los canales de GPIO
```

Anexo VIII: Código maestro lectura HC-SR04 y TS, TSultrasonido.py

```
#######################################################
#         PROYECTO FINAL DE CARRERA 2020              #
#######################################################
# Titulo: Desarrollo e implementacion de una red de #
# de sensores para monitorizacion basada en ZigBee  #
#######################################################
# Nombre:  TSultrasonido.py                           #
# Autor:   Juan Manuel Marquez Flores                 #
# Carrera: Grado en Ingenieria Electromecanica        #
# Entidad: Universidad Loyola Andalucia               #
#######################################################
import httplib
import urllib
import time
import RPi.GPIO as GPIO
# Numeracion BCM
GPIO.setmode(GPIO.BCM)
# Declaracion de pines GPIO
TRIG = 4    #Pin 6 en Coocking Hacks shield
ECHO = 25   #Pin 5 en Coocking Hacks shield
# Establece si es canal de salida o entrada
GPIO.setup(TRIG, GPIO.OUT)
GPIO.output(TRIG, False)
GPIO.setup(ECHO, GPIO.IN)
# Clave unica para subir a canal de ususario de ThingSpeak
key = "6EAM55ROJ4R4L319"
# Se define la funcion de ultrasonido
def ultrasonido():
    while True: # Ciclo infinito, para terminarlo pulsar Ctrl+C
        time.sleep(2) # Tiempo de espera para estabilizar el sensor
        GPIO.output(TRIG, True)
        time.sleep(0.00001) # Tras ser encendido, se apaga de nuevo
        GPIO.output(TRIG, False)
        # El sensor envia el pulso e inicia el cronometro
        while GPIO.input(ECHO) == 0:
            pass
            inicio_pulso = time.time()
        # Recoge el pulso y para el cronometro
        while GPIO.input(ECHO) == 1:
            pass
            fin_pulso = time.time()
        # El tiempo recorrido es la diferencia entre inicio y final
        duracion_pulso = fin_pulso - inicio_pulso
        distancia = duracion_pulso * 17150 # El numero es la Vel Sonido
entre 2
```

```
        dist= round(distancia, 2) # Se redondea a 2 decimales
        # Comienza el proceso de conexion con ThingSpeak
        params = urllib.urlencode({'field4': dist, 'key':key })
        headers = {"Content-typZZe": "application/x-www-form-
urlencoded","Accept": "text/plain"}
        conn = httplib.HTTPConnection("api.thingspeak.com:80")
        # Intenta publicar el dato haciendo peticion al servidor
        try:
            conn.request("POST", "/update", params, headers)
            response = conn.getresponse()
            print ("Distancia medida = %.2f cm" % dist)
            print (response.status, response.reason)
            data = response.read()
            conn.close()
        except:
            print ("Conexcion fallida")
        break
# Salvo que se interrumpa por teclado o haya error, no se interrumpira el
bucle
if __name__ == ("__main__"):
    try:
        while True:
            ultrasonido()
            time.sleep(15)
    except KeyboardInterrupt:
        print("Programa cancelado por el usuario")
        GPIO.cleanup() # Libera los canales de GPIO
```

7. Bibliografía

Acharya, A. D., & Patil, S. N. (2020). IoT based Health Care Monitoring Kit. *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)* (págs. 363-368, doi: 10.1109/ICCMC48092.2020.ICCMC-00068). Erode: IEEE.

Albatross. (7 de Enero de 2014). *Cplusplus*. Obtenido de Cplusplus: https://www.cplusplus.com/info/description/

Aprendiendo Arduino. (2017). *Aprendiendo Arduino*. Obtenido de Aprendiendo Arduino: https://aprendiendoarduino.wordpress.com/2017/07/09/i2c/

Arduino. (2020). *Arduino*. Obtenido de Arduino: https://www.arduino.cc/

aREST. (2020). *aREST*. Obtenido de AREST: https://arest.io/

Baert, M., Rossey, J., Shahid, A., & Hoebeke, J. (2018). The Bluetooth Mesh Standard: An Overview and Experimental Evaluation. *Sensors, 18(8)*, 1-24, doi:10.3390/s18082409.

Blac Solutions. (30 de Junio de 2015). *Blac Solutions*. Obtenido de Blac Solutions: https://blacsol.com/iot-internet-of-things

Bluetooth. (23 de Junio de 2020). *Bluetooth*. Obtenido de Bluetooth: https://www.bluetooth.com/

Cilfone, A., Davoli, L., Belli, L., & Ferrari, G. (2019). Wireless Mesh Networking: An IoT-Oriented. *Future Internet, 11(4)*, 1-35, doi:10.3390/fi11040099.

Coocking Hacks. (13 de Enero de 2020). *Coocking Hacks*. Obtenido de Coocking Hacks: https://www.cooking-hacks.com/documentation/tutorials/raspberry-pi-to-arduino-shields-connection-bridge.html

Dávila, M. A., Pérez, J. F., Mantilla, W., & Moreno, J. E. (2016). Diseño de una Red Inalámbrica con Tecnología ZigBee para la Implementación de un Sistema Domótico. *Revista de Ciencia e Ingeniería del Instituto Tecnológico Superior de Coatzacoalcos, 3(3)*, 415-420.

del Valle, L. (21 de Marzo de 2017). *Programar fácil*. Obtenido de Programar fácil: https://programarfacil.com/blog/arduino-blog/sensor-dht11-temperatura-humedad-arduino/

del Valle, L. (25 de Mayo de 2020). *Programar facil*. Obtenido de Programar facil: https://programarfacil.com/podcast/proyectos-iot-con-arduino/

Digi. (11 de Junio de 2018). *Digi*. Obtenido de Digi: https://www.digi.com/resources/documentation/digidocs/90001458-13/default.htm

Digi. (29 de Noviembre de 2019). *Xbee*. Obtenido de Xbee: https://xbee.cl/que-es-xbee/

Electronica Estudio. (13 de Marzo de 2020). *Electronica Estudio*. Obtenido de Electronica Estudio: https://www.electronicaestudio.com/que-es-un-microcontrolador/

Fialho, V., & Azevedo, F. (2018). Wireless Communication Based on Chirp Signals for LoRa IoT Devices. *ISEL Academic Journal of Electronics, Telecommunications and Computers, 4(1)*, 1-5, doi: 10.34629/ipl.isel.i-ETC.51.

Fritzing. (2020). *Download: Fritzing*. Obtenido de Fritzing: https://fritzing.org/download/

Geany. (2020). *Geany*. Obtenido de Geany: https://www.geany.org/

Google. (2020). *Google Cloud IoT*. Obtenido de Google Cloud IoT: https://cloud.google.com/solutions/iot/

Gracia, L. (28 de Abril de 2013). *Un poco de Java*. Obtenido de Un poco de Java: https://unpocodejava.com/2013/04/28/que-es-coap/

Gracia, M. (1 de Agosto de 2019). *Deloitte Spain*. Obtenido de Deloitte Spain: https://www2.deloitte.com/es/es/pages/technology/articles/IoT-internet-of-things.html

Güven, Y., Cosgun, E., & Kocaoglu, S. (2017). Understanding the Concept of Microcontroller Based Systems To Choose The Best Hardware For Applications. *International Journal of Engineering And Science, 6(9)*, 38-44.

Hard Kernel. (2020). *Hardkernel*. Obtenido de Hardkernel: https://www.hardkernel.com/

Hetpro. (25 de Noviembre de 2017). *Hetpro*. Obtenido de Hetpro: https://hetpro-store.com/TUTORIALES/arduino-lm35/

IBM. (2020). *Internet of Things: IBM*. Obtenido de IBM: https://www.ibm.com/internet-of-things

Intel. (2020). *Intel*. Obtenido de Intel: https://www.intel.es

Libelium. (2020). *Libelium*. Obtenido de Libelium: http://www.libelium.com/

Libre Computer. (2020). *Libre Computer*. Obtenido de Libre Computer: https://libre.computer/products/boards/aml-s905x-cc/

Linux. (2020). *Linux*. Obtenido de Linux: https://www.linux.org/

Llamas, L. (2015). *Luis Llamas*. Obtenido de Luis Llamas: https://www.luisllamas.es/salidas-analogicas-pwm-en-arduino/

Llamas, L. (16 de Junio de 2015). *Luis Llamas*. Obtenido de Luis Llamas: https://www.luisllamas.es/medir-distancia-con-arduino-y-sensor-de-ultrasonidos-hc-sr04/

Llamas, L. (17 de Abril de 2019). *Luis Llamas*. Obtenido de Luis Llamas: https://www.luisllamas.es/que-es-mqtt-su-importancia-como-protocolo-iot/

LoRa Alliance. (24 de Junio de 2020). *LoRa Alliance*. Obtenido de LoRa Alliance: https://lora-alliance.org/

Mahzan, N. N. (2020). A Design of Smart IoT-Based College Room Using Arduino. *Journal of Physics: Conference Series, 1529(1)*, 1-10, doi:10.1088/1742-6596/1529/2/022045.

MathWorks. (2020). *MathWorks*. Obtenido de MathWorks: https://es.mathworks.com/

MDN. (29 de Mayo de 2020). *MDN web docs*. Obtenido de MDN web docs: https://developer.mozilla.org/es/docs/Web/HTTP/Overview

Mendieta, T. P., Herrera, J., & Jimenez Peña, A. (2019). La Capacidad del IOT de Transformar el Futuro. *Revista Avenir, 1(1)*, 15-18.

Node-RED. (2020). *Node-RED*. Obtenido de Node-RED: https://nodered.org/

NXP. (2020). *NXP*. Obtenido de NXP: https://www.nxp.com

Oracle. (26 de Junio de 2020). *Oracle*. Obtenido de Oracle: https://www.oracle.com/es/internet-of-things/what-is-iot.html

Osiberia. (23 de Diciembre de 2016). *OpenSource Osiberia*. Obtenido de OpenSource Osiberia: https://osiberia.org/que-es-una-red-iot-lpwan/

Oyarce, A. (2008). *Guía del Usuario, Xbee Series 1.* Santiago: Ingeniería MCI LTDA.

Python. (2020). *Python*. Obtenido de Python: https://www.python.org/

Raspberry Pi. (2020). *Donwload: Noobs*. Obtenido de Raspberry Pi: https://www.raspberrypi.org/downloads/noobs/

Raspberry Pi. (2020). *Raspberry Pi*. Obtenido de Raspberry Pi: https://www.raspberrypi.org

Raspbian. (2020). *Raspbian*. Obtenido de Raspbian: https://www.raspbian.org/

Redeweb. (17 de Enero de 2019). *Redeweb*. Obtenido de Redeweb: https://www.redeweb.com/articulos/bluetooth-mesh-permite-aplicaciones-completamente-nuevas/

Rogelio, L. (3 de Junio de 2015). *Botboss*. Obtenido de Botboss: https://botboss.wordpress.com/2015/06/03/programar-gpio-con-raspberry-pi-2-y-python/

Ruiz, L. (11 de Noviembre de 2019). *Miscelánea: Psicología y mente*. Obtenido de Psicología y mente: https://psicologiaymente.com/miscelanea/tipos-de-sensores

Sahitya, G., Balaji, N., Naidu, C. D., & Abinaya, S. (2017). Designing a Wireless Sensor Network for Precision Agriculture Using Zigbee. *IEEE 7th International Advance Computing Conference (IACC)* (págs. 287-291, doi: 10.1109/IACC.2017.0069). Hyderabad: IEEE.

SIgFox. (24 de Junio de 2020). *SIgFox*. Obtenido de SIgFox: https://www.sigfox.com/en

Sparkfun. (2020). *Productos: Sparkfun*. Obtenido de Sparkfun: https://www.sparkfun.com/products/11812

ThingSpeak. (2020). *ThingSpeak*. Obtenido de ThingSpeak: https://thingspeak.com/

Thonny. (2020). *Thonny*. Obtenido de Thonny: https://thonny.org/

Tschofenig, H., Arkko, J., Thaler, D., & McPherson, D. (2015). Architectural Considerations in Smart Object Networking. *Request for Comments, 7452*, 1-24, doi: 10.17487/RFC7452.

Twitter. (2020). *Twitter*. Obtenido de Twitter: https://twitter.com/explore

Villagómez, C. (13 de Febrero de 2018). *CommentÇaMarche*. Obtenido de CommentÇaMarche: https://es.ccm.net/contents/789-introduccion-a-wifi-802-11-o-wifi

VNC Viewer. (2020). *VNC Viewer*. Obtenido de VNC Viewer: https://www.realvnc.com/es/connect/download/viewer/

Wi-Fi Alliance. (24 de Junio de 2020). *Wi-Fi Alliance*. Obtenido de Wi-Fi Alliance: https://www.wi-fi.org/discover-wi-fi/specifications

World Health Organization. (15 de Marzo de 2020). *World Health Organization*. Obtenido de World Health Organization: https://www.who.int/health-topics/cardiovascular-diseases/#tab=tab_1

Zetta. (2020). *Zetta*. Obtenido de Zetta: https://www.zettajs.org/

ZigBee Alliance. (24 de Junio de 2020). *ZigBee Alliance*. Obtenido de ZigBee Alliance: https://zigbeealliance.org/